Titre: Title:	Compression vidéo par compensation du mouvement utilisant des transformations adaptatives
Auteur: Author:	Zafer Diab
Date:	1997
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Diab, Z. (1997). Compression vidéo par compensation du mouvement utilisant des transformations adaptatives [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8983/

UNIVERSITÉ DE MONTRÉAL

COMPRESSION VIDÉO PAR COMPENSATION DU MOUVEMENT UTILISANT DES TRANSFORMATIONS ADAPTATIVES

ZAFER DIAB
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉLECTRIQUE)
AVRIL 1997

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE

Ce mémoire intitulé:

COMPRESSION VIDÉO PAR COMPENSATION DU MOUVEMENT UTILISANT DES TRANSFORMATIONS ADAPTATIVES

présenté par: DIAB Zafer

en vue de l'obtention du diplôme de: Maître ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de:

M. BRAULT Jean Jules, Ph.D., président

M. COHEN Paul, Ph.D., membre et directeur de recherche

M. NERGUIZIAN Chahé, M.Ing., membre

A ma chère maman

Remerciements

Je tiens tout d'abord à remercier M. Paul Cohen pour son aide et ses judicieux conseils tout au cours de ce projet de recherche. Merci également à la chaire CRSNG-Noranda pour leur support financier.

Un grand merci à Hai et à Geneviève, de même qu'à mes collègues du Groupe de Recherche en Perception et Robotique pour leur aide et leur patience, en particulier Christine Frobert pour son aide précieuse dans la rédaction de ce document.

Je remercie mes parents Souheil et Fadia de m'avoir donné tout l'amour qu'un enfant peut espérer et mon frère Maher de m'avoir soutenu dans les moments difficiles.

Enfin, et surtout, merci à ma fiancée Sawsan pour son support moral. Son amour m'a toujours motivé pour travailler fort.

Résumé

Dans les techniques classiques de codage par compensation de mouvement, on estime le mouvement de chaque bloc d'une image par un vecteur. Cette approximation grossière se manifeste par un effet de bloc visible pour les applications à bas débit. Dans ce mémoire, nous proposons une méthode de codage utilisant une meilleure description du mouvement des blocs basée sur la représentation du mouvement dans un domaine transformé. Les bases de transformation sont apprises et adaptées de manière autonome en utilisant un réseau de neurones à apprentissage Hebbien. Ceci a l'avantage de limiter l'erreur de reconstruction tout en gardant une représentation compacte du champ de mouvement.

Une étude spatiale sur les différences entre l'image courante et l'image reconstruite à l'instant précédent permet de diviser l'image en régions dynamiques et statiques. Le flux optique est calculé dans les régions dynamiques et divisé en blocs de mouvement. Par la suite, le mouvement dans chaque bloc dynamique est transformé et quantifié. Le choix de la matrice de transformation de chaque bloc dépend de la classe de mouvement bidimensionnel présent dans le bloc.

Un nombre fixe de classes de mouvement est utilisé. Nous associons à chacune de ces classes une transformation en composantes principales de l'amplitude et de la phase du champ de mouvement. Les bases de transformation de chaque classe sont apprises en utilisant un réseau de neurones à apprentissage Hebbien. Chaque bloc de mouvement est attribué à la classe dont la composante principale donne le plus grand produit scalaire avec ce bloc. Puisque les coefficients de transformation sont décorrélés et classés par ordre décroissant de variance, nous pouvons obtenir une bonne approximation du mouvement en gardant un nombre restreint de coefficients.

La transformation associée à chaque classe de mouvement (et par conséquent la nature de chaque classe) est adaptée tout au long du processus de codage pour refléter la structure particulière du mouvement présent. L'adaptation est établie en se basant sur les mouvements déjà reconstruits car l'information d'adaptation est coûteuse à transmettre. Puisque les matrices de transformation sont orthonormales, la reconstruction du champ de mouvement revient tout simplement à une multiplication par la transposée de la matrice de transformation.

La reconstruction des images est faite par interpolation des déplacements. L'avantage de cette méthode est qu'elle se base sur la douceur du champ de vitesses par opposition aux méthodes classiques se basant sur la douceur des intensités des images, ce qui se traduit par une erreur de reconstruction moindre. Les erreurs de reconstruction sont souvent localisées aux occlusions et causent une accumulation de l'erreur dans le temps. Nous limitons cette erreur en transmettant des images de mise à jour codées

dans le domaine spatial à chaque fois que l'erreur devient trop grande. Ces images sont transmises uniquement pour les blocs dynamiques et sont codées en gardant un nombre fixe de coefficients dans le domaine transformé.

Abstract

Motion compensated compression strategies usually involve estimating the motion of each image block by a single vector. However, this coarse approximation usually results in visible block effects for low bit-rate applications. In this thesis, we propose a coding scheme based on a more accurate description of the motion present in each block by representing it in a transformed domain. The transformation basis is learned and adapted in a self-organized manner to represent the motion of the scene. This has the advantage of limiting reconstruction errors while achieving a compact motion description.

A spatial study on the differences between the incoming image and the previously reconstructed image allows us to divide the image into dynamic and static regions. The optical flow in the dynamic regions is first computed and subdivided into motion blocks. The motion in each dynamic block is then transformed and quantized. The choice of the transformation for each block depends upon the class of the 2D motion present in the block.

A fixed number of motion classes is adopted. A principal component transform of

the module and the phase of the image motion is associated to each class. A neural network using the Hebbian learning rule is used to learn the principal components of the motion of each class. Each particular motion block is associated with the class having the greatest scalar product between its principal component and the block. Since the transform coefficients are uncorrelated and ordered in descending order of variances, retaining the first few coefficients provides an adequate approximation of the input motion and achieves good compression.

The transforms associated to each motion class (and therefore the nature of each class) are adapted during the coding process, in order to reflect the particular structure of the motion present. Since the adaptation information is costly to transmit, it is established on the basis of previously transmitted and reconstructed motion flows. Owing to the orthonormal nature of the transforms involved, the reconstruction of the image motion flow simply involves multiplication by the transpose of the forward transform matrices.

Image reconstruction is made by linear patch displacement interpolation. This method gives improved results since it assumes smoothness in the velocities rather than in image intensities. Reconstruction errors tend to be mostly located at object boundaries and cause an error accumulation in time. We attenuate this error by transmitting intra-frame images each time this error becomes large. These images are transmitted uniquely for dynamic blocks and are coded by retaining a fixed number of coefficients in the transformed domain.

Table des matières

Liste des figures

Liste des tableaux

Liste des notations

$[I_t(x,y)]$ intensité lumineuse du pixel (x,y) de l'image au temps t:

$[\hat{I}_t(x,y)]$ intensité lumineuse reconstruite du pixel (x,y) au temps t:

$[\vec{v}_t(x,y) = (v_{t_1}(x,y), v_{t_2}(x,y))]$ vecteur de mouvement calculé entre l'image au temps t et l'image au temps $t+1$ au pixel (x,y):

$[a_t(x,y)]$ et $[\theta_t(x,y)]$ amplitude et phase du vecteur de mouvement calculé entre l'image au temps t et l'image au temps $t+1$ au pixel (x,y):

$[\tilde{a}]$ et $[\tilde{\theta}]$ amplitude et phase du mouvement d'un bloc représentés par des vecteurs correspondant aux lignes du bloc plaçées l'une à la suite de l'autre:

$[T^a_{c_a}(i)]$ et $[T^\theta_{c_\theta}(i)]$ $i^{\text{ème}}$ composante principale de l'amplitude et de la phase de la classe c_a et c_θ respectivement:

$[\tilde{T}^{*a}_{c_a}]$ et $[\tilde{T}^{*\theta}_{c_\theta}]$ première composante principale de l'amplitude et de la phase de la classe c_a et c_θ respectivement représent'ee en forme de matrice:

$[y^a_{c_a}(i)]$ et $[y^\theta_{c_\theta}(i)]$ $i^{\text{ème}}$ coefficient de transformation de l'amplitude et de la phase du mouvement d'un bloc selon la classe c_a ou c_θ respectivement.

Liste des annexes

Introduction

Le codage de séquences d'images digitales constitue un domaine de recherche actuellement en plein essor. Une grande compression d'images peut permettre leur transmission dans des canaux à bas débit et par conséquent une utilisation efficace des canaux de transmission actuellement accessibles. Elle joue, donc, un rôle primordial dans des domaines aussi variés que la télévision, la vidéo-conférence, la télémédecine, l'industrie des divertissements (jeux vidéos), le multimédia ou le contrôle télé-robotique.

Le problème majeur réside dans le fait que la transmission d'images dans leur format naturel exige des débits prohibitifs et qu'un effort tout particulier doit être mis à réduire ces débits sans dégrader l'information transmise de façon inacceptable. Dans cette optique, le but de la compression est de représenter les séquences d'images sous la forme la plus compacte possible tout en assurant que les dégradations introduites ne seront pas gênantes visuellement pour l'application considérée. Cette compression est généralement possible car une séquence d'images contient de l'information répétitive appelée redondance (Bhaskarau et Koustantinides 1995).

Une première source de redondance réside dans la structure intra-image du signal. En effet, la luminance d'un pixel est souvent corrélée avec la luminance des pixels adjacents (Zschunke 1977). L'exploitation de cette redondance, dite spatiale, consiste à prédire la luminance d'un pixel à partir de celles de ses voisins. La deuxième source de redondance présente dans une séquence d'images provient de la corrélation entre la luminance du même point au cours du temps (Ishiguro et Inuma 1982). En effet, dans des images successives dans le temps, on peut retrouver les mêmes objets à la même position ou déplacés. L'exploitation de cette redondance, dite temporelle, consiste à estimer le déplacement des pixels entre plusieurs images successives d'une séquence et à prédire les intensités lumineuses dans une image à partir des intensités de l'image précédente compensées selon ce mouvement.

A part la répétition d'information, de la compression supplémentaire peut être obtenue en exploitant les limites des capacités perceptuelles du système visuel humain (Rajala, Civanlar et Lee 1988). Ainsi, certaines informations n'ont pas besoin d'être transmises puisque, de toute façon, elles ne seront pas discernées en raison de la résolution spatiale ou temporelle limitée du système visuel humain.

La compensation de mouvement est la méthode la plus utilisée pour la compression vidéo. La plupart des approches de compensation de mouvement proposées dans la littérature sont basées sur une subdivision de l'image en blocs et l'estimation du mouvement avec un seul vecteur par bloc (Jain et Jain 1981). Ceci revient à supposer que le mouvement de la portion de la scène projetée sur ce bloc est translationnel

et fronto-parallèle (Burger et Bhanu 1992). Cette hypothèse se manifeste par une erreur de reconstruction élevée qu'on transmet pour obtenir une qualité acceptable des images reconstruites au récepteur. Sachant que les erreurs de reconstruction sont concentrées aux occlusions, le signal d'erreur, étant localisé en hautes fréquences, ne peut pas être compressé efficacement. Ceci rend la compensation du mouvement par bloc inappropriée pour les applications à bas débit (Li, Lundmark et Forchheimer 1994).

Dans ce mémoire, nous proposons de se baser sur le mouvement dense estimé par le flux optique pour faire la compensation de mouvement dans des séquences d'images monochromes. La compression du champ de mouvement dense peut apparaître comme un défi. En effet, ce champ est composé d'un vecteur par pixel et comporte par conséquent le double de la quantité d'information contenue dans les images originales. Par contre, en raison de la continuité du mouvement dans le temps, ce champ est très redondant et peut, par conséquent, être représenté de façon compacte. Le mouvement est modélisé dans un espace transformé comme expliqué au chapitre 3. Cette modélisation est adaptée dans le temps en utilisant un réseau de neurones à apprentissage Hebbien pour orienter les bases de transformation selon les directions principales du mouvement.

Les paramètres du modèle sont quantifiés et transmis au récepteur. Ce dernier reconstruit le champ de mouvement dense et l'utilise pour interpoler les intensités lumineuses des images reconstruites.

Les erreurs dans le calcul du flux optique, la modélisation de ce dernier et les interpolations pour la reconstruction des images causent une accumulation inévitable d'erreur dans les images reconstruites au récepteur. Nous limitons ces erreurs en transmettant une image de mise à jour lorsque cette erreur devient trop grande.

L'utilisation du mouvement dense pour faire la compensation de mouvement a permis d'obtenir des taux de compression suffisemment élevés pour fonctionner à bas débit. Ceci est causé par la continuité du mouvement dans le temps qui nous a permis de représenter le mouvement dense avec autant d'information qu'un déplacement par bloc.

Le mémoire est organisé comme suit. Le chapitre 1 présente une revue bibliographique sur les méthodes de compensation du mouvement. La structure générale du système de compensation du mouvement proposé est présentée au chapitre 2. Le chapitre 3 décrit la modélisation du champ de mouvement en sources composées utilisée par notre approche. Le chapitre 4 décrit la méthode de codage que nous avons choisie en décomposant ses différents modules et en étudiant en détails tous les paramètres qu'il utilise. L'ensemble des résultats expérimentaux ainsi qu'une comparaison de performance avec MPEG, à partir de tests d'évaluation de qualité objectifs et subjectifs, est présenté dans le chapitre 5. Enfin, une conclusion vient résumer l'ensemble du travail effectué en décrivant les caractéristiques de notre système ainsi que les résultats obtenus et propose des éventuels développements en vue d'améliorer notre méthode.

Chapitre 1

Revue bibliographique

Notre travail touche au codage des séquences d'images par compensation du mouvement. Ce chapitre dresse un aperçu des travaux réalisés dans ce domaine et souligne les liens qui les unissent à notre recherche. La compensation de mouvement est la méthode la plus populaire pour la compression des séquences d'images. Elle permet de réduire l'information essentielle à la présentation des images en éliminant les redondances temporelles. Nous commençons par introduire à la section 1.1 le principe du codage par compensation de mouvement. Les prédictions effectuées en compensation de mouvement sont basées sur le mouvement projeté dans le domaine des images. La section 1.2 présente les systèmes les plus utilisés pour estimer ce mouvement.

Les systèmes de compensation de mouvement existant peuvent être classés dans plusieurs catégories. Dans la section 1.3, nous présentons ces différents systèmes. La section 1.4 effectue une analyse critique des approches de compensation de mouvement

existants et présente. dans ses principes. notre approche pour la compression des séquences.

1.1 Principe de la compensation de mouvement

L'une des approches pour faire la compression des séquences d'images repose sur la prédiction spatiale qui consiste à prédire la luminance d'un pixel à partir des valeurs de luminance dans un voisinage spatial de ce pixel. La faiblesse de cette approche est que la corrélation entre la luminance à prédire et les luminances dans le voisinage est limitée au niveau des contours présents dans chaque image. En effet. deux objets voisins n'ont pas nécessairement des luminances ressemblables.

Une alternative consiste à faire la prédiction temporellement en tenant compte du mouvement. La luminance d'un pixel, en compensation du mouvement. est estimé donc par la luminance du même pixel dans l'image précédente. Cette hypothèse est la base des méthodes d'estimation du mouvement expliquées à la section 1.2 et s'appelle la contrainte du gradient. La justification de cette hypothèse est que la corrélation est grande le long des trajectoires de mouvement. La figure 1.1 montre le schéma global des systèmes de codage des séquences d'images par compensation de mouvement. La luminance d'un pixel au temps $t+1$, $\widetilde{I}\ (t+1)$, est prédite par sa luminance reconstruite à l'image précédente $\hat{I}\ (t)$. La position du pixel est trouvé en se basant sur les vecteurs de mouvement $\widetilde{v}\ (t+1)$ estimé entre l'image au temps t et l'image au temps $t+1$. L'erreur de prédiction E est calculé comme étant la

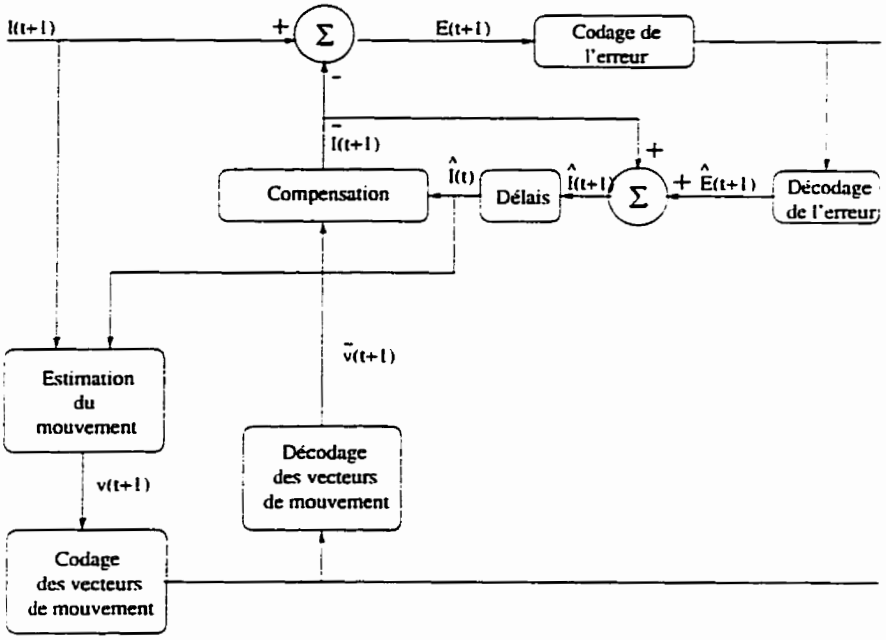


Figure 1.1 : Système typique de codage vidéo par compensation de mouvement

différence pixel par pixel entre $\widetilde{I}\ (t+1)$ et $I(t+1)$. Les vecteurs de mouvement et l'erreur de reconstruction sont codés et transmis au récepteur. Le plus grand défi dans ces méthodes est le codage des vecteurs de mouvement requis pour obtenir la compression voulue avec une qualité acceptable des images.

1.2 Estimation du mouvement

Quand des objets bougent devant une caméra ou qu'une caméra se déplace dans un environnement statique, nous pouvons associer à chaque point de la scène sa vitesse physique. Par exemple, dans la figure 1.2, le point P_0 s'est déplacé au point P_1 pendant l'intervalle de temps dt, nous pouvons donc associer au point P_0 la vitesse $\vec{V}$.

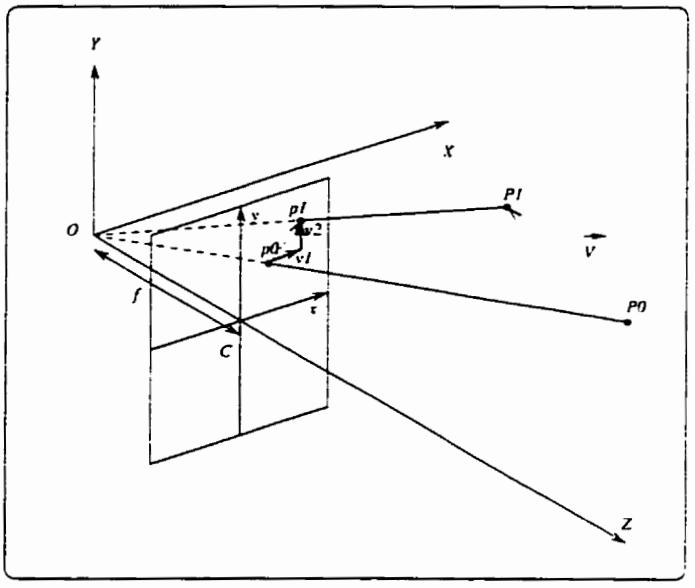


Figure 1.2 : Projection des vitesses dans l'image

Nous pouvons associer à chaque point de l'image la projection de cette vitesse sur l'image, ce qui donne le champ de mouvement en chaque point $p_0(x, y)$ soit $\vec{v} = (v_1, v_2)$. L'équation de cette projection est (Horn 1986):

$$(v_1(x,y), v_2(x,y)) = \vec{v} = f\frac{(\vec{P_0} \times \vec{V}) \times \vec{Z}}{(\vec{P_0}.\vec{Z})^2} \tag{1.1}$$

où f est la distance focale de la caméra, c'est-à-dire la distance entre le centre C et le foyer de la lentille O.

Le champ de mouvement est un concept purement géométrique. En effet, nous ne pouvons pas calculer ce champ de vecteurs car les mouvements dans la scène ne nous sont pas connus et seules les intensités lumineuses dans l'image sont connues.

Le mouvement dans une image peut être représenté par le flux optique. Ce dernier

correspond au mouvement apparent des patrons d'intensité lumineuse. Le flux optique constitue une approximation du vrai champ de mouvement.

Idéalement, le flux optique va correspondre au champ de mouvement, mais ceci n'est pas toujours le cas. Par exemple, une sphère uniforme qui tourne autour de son axe ne cause pas de changements dans les intensités lunineuses, pourtant son mouvement est non nul. Par contre, si cette sphère est fixe, mais illuminée par une source de lumière qui bouge, on aura des mouvements dans les patrons d'intensités lumineuses de la sphère, pourtant son mouvement est nul. Dans la plupart des cas, cependant, le flux optique est une bonne approximation du champ projeté.

Plusieurs méthodes pour calculer le flux optique ont été proposées et d'autres continuent à apparaître. Les méthodes les plus souvent mentionnées dans la littérature sont les techniques différentielles qui se basent sur la contrainte du gradient. Cette contrainte est obtenue en supposant que les pixels ne changent pas d'intensité $I(x, y)$ entre deux instants successifs et peut être exprimée selon l'équation (Horn et Schunck 1981):

$$\vec{\nabla} I_t(x, y).\vec{v} + \frac{dI_t(x, y)}{dt} = 0 \tag{1.2}$$

Pour calculer les dérivées temporelles des intensités lumineuses de l'image, on doit supposer que ce signal est différentiable. Ceci nous oblige de réaliser un lissage des images avant de pouvoir calculer le flux optique.

L'équation du gradient est à deux variables (v_1 et v_2) et donc elle nous permet de calculer le vecteur de déplacement seulement dans la direction du gradient d'intensité.

Ceci s'appelle le problème d'ouverture. Pour résoudre ce problème, nous avons besoin d'une contrainte additionnelle qu'on peut obtenir en se basant sur des différentiations d'ordre deux (Nagel 1983, Nagel et Enkelmann 1986, Nagel 1987, Demicheli, Sandini, Tistarelli et Torre 1988), en imposant des contraintes de lissage sur les voisinages (Lucas et Kanade 1981) ou en faisant du lissage global (Horn et Schunck 1981).

Le champ de mouvement contient le double de la quantité d'information présente dans les images (1 vecteur/pixel). Pour compresser le volume d'information contenu dans ce dernier, plusieurs modèles ont été proposés. Dans ce qui suit, nous décrirons ces modèles et nous présentons les mesures utilisées pour évaluer la qualité de ces modèles.

1.2.1 Modélisation du mouvement

Plusieurs modèles ont été proposés pour décrire la projection du mouvement rigide d'un objet sur le plan image. Pour que ces modèles soient applicables, il faut segmenter les images en régions subissant des mouvements rigides. L'information de segmentation et les paramètres du modèle pour chacune des régions constituent la description du mouvement.

Le modèle de mouvement est une équation qui relie les points (x, y) de la région aux points déplacés (x', y'). Au tableau 1.1, nous avons plusieurs modèles qui sont souvent utilisés dans la littérature. Le modèle homographique (Tsai et Huang 1981) est celui qui décrit la projection du vrai mouvement sur le plan image (voir la section A.1 pour

la démonstration). Le bruit présent sur les images cause que les paramètres de ce modèle sont difficiles à estimer en pratique. Par conséquent, les modèles polynomiaux sont utilisés à cause de la simplicité de l'estimation de leurs paramètres. Parmis tous les modèles présentés dans le tableau 1.1, le modèle translationnel à 2 paramètres est celui qui est le plus souvent utilisé. Il constitue la base des techniques de compensation du mouvement par bloc (section 1.3.1) et par régions polygonales (section 1.3.2.1).

1.2.2 Evaluation de la qualité du modèle

Plusieurs mesures sont possibles pour évaluer la qualité d'un mouvement calculé à partir d'un modèle en tant qu'approximation du champ de mouvement projeté. Lin et Barron (1994) proposent les deux suivantes:

1. **Erreur d'angle** Cette mesure présuppose qu'on connait le champ de mouvement ($\vec{v}(x,y)\ \forall(x,y)$). L'erreur d'angle ($\Phi(x,y)$) est alors une mesure de distance entre les vecteurs de ce champ et les vecteurs du mouvement calculé ($\hat{\vec{v}}$). Cette mesure tient compte de l'amplitude et de la phase de ces vecteurs comme illustré à la figure 1.3. Le calcul est donné par les équations suivantes (Fleet et Jepson 1990):

$$\Phi(x,y) = \cos^{-1} < \vec{v_N}(x,y), \hat{\vec{v}}_N(x,y) > \quad (1.10)$$

$$\vec{v_N} = \frac{(v_1, v_2, 1)^T}{\|\vec{v_N}\|},\ \hat{\vec{v}}_N = \frac{(\hat{v_1}, \hat{v_2}, 1)^T}{\|\hat{\vec{v}}_N\|} \quad (1.11)$$

Tableau 1.1 : Modèles de mouvement

Modèle	Equation
Modèle homographique à 8 paramètres	$x' = \frac{a_1x + a_2y + a_3}{a_7x + a_8y + 1}$ $y' = \frac{a_4x + a_5y + a_6}{a_7x + a_8y + 1}$ (1.3)
Modèle polynômial à 12 paramètres	$x' = a_1x^2 + a_2x + a_3xy + a_4y^2 + a_5y + a_6$ $y' = b_1x^2 + b_2x + b_3xy + b_4y^2 + b_5y + b_6$ (1.4)
Modèle bilinéaire à 8 paramètres	$x' = a_1xy + a_2x + a_3y + a_4$ $y' = b_1xy + b_2x + b_3y + b_4$ (1.5)
Modèle affine à 6 paramètres	$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x\cos\theta_x & -S_y\sin\theta_y \\ S_x\sin\theta_x & S_y\cos\theta_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$ (1.6)
Modèle affine à 4 paramètres	$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S & -\theta \\ \theta & S \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$ (1.7)
Modèle translationnel à 2 paramètres	$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$ (1.8)
Modèle nul (0 paramètres)	$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}$ (1.9)

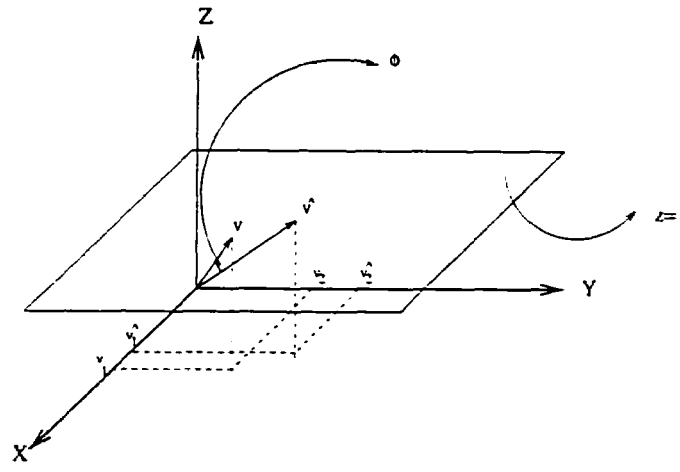


Figure 1.3 : Erreur d'angle

où $< ... >$ désigne le produit scalaire.

2. **Erreur quadratique moyenne de reconstruction** (EQM) Si nous ne connaissons pas le vrai mouvement, nous utilisons le mouvement calculé pour reconstruire l'image $\hat{I}_{t+1}$ et nous nous basons sur l'erreur quadratique moyenne de reconstruction (EQM) pour évaluer la qualité du modèle. Cette erreur est donnée selon l'équation suivante:

$$EQM = \sqrt{\frac{\sum_x \sum_y (I_{t+1}(x,y) - \hat{I}_{t+1}(x,y))^2}{MAX_X \times MAX_Y}} \tag{1.12}$$

Lin et Barron (1994) ont testé ces deux mesures sur des séquences synthétiques[1] et ont trouvé qu'elles sont corrélées entre elles. Ceci permet de tirer les deux conclusions suivantes:

[1] Dans lesquelles on connaît le vrai mouvement a priori

1. Nous pouvons nous fier à l'EQM pour tirer des conclusions sur la qualité du mouvement calculé. En d'autres termes, pour évaluer la qualité d'un mouvement quand on n'a pas le champ de mouvement projeté, il suffit d'utiliser le mouvement calculé pour reconstruire les images et comparer les images reconstruites avec les images originales.

2. Nous pouvons comparer la qualité des images reconstruites en utilisant plusieurs modèles de mouvement en se basant sur les erreurs d'angle entre les mouvements calculés selon ces modèles, sans avoir à effectuer directement la reconstruction d'images.

1.3 Compensation de mouvement

Pour coder efficacement une séquence d'images, il faut tenir compte des redondances entre les images successives. La technique la plus souvent utilisée pour réaliser ceci est la compensation de mouvement.

Les méthodes de compensation de mouvement qu'on trouve dans la littérature peuvent être classées de plusieurs manières. On peut, alors, se baser sur la modélisation du mouvement pour distinguer entre les systèmes se basant sur un des modèles de la section 1.2.1 et ceux se basant sur un flux optique dense. Le mouvement peut être transmis au récepteur ou estimé chez ce dernier par des méthodes récursives ou en se basant sur les images déjà reconstruites.

Nous pouvons aussi distinguer les différents systèmes selon le domaine sur lequel ils considèrent le mouvement. En effet, plusieurs systèmes considèrent le mouvement dans des régions fixes, comme des blocs par exemple. D'autres segmentent les images en régions de formes arbitraires. Enfin, le mouvement peut être vu comme un champ bidimensionnel complexe (un vecteur par pixel) défini sur l'image au complet.

1.3.1 Compensation de mouvement par bloc

L'ensemble de ces techniques présuppose que le mouvement est localement modélisable par une simple translation (section A.1.6) entre deux instants. Pour calculer alors cette translation, on divise d'abord l'image à coder en blocs égaux. L'estimation du mouvement est faite à l'aide de méthodes d'appariement de blocs qui visent à minimiser une mesure de similarité entre ces blocs et des blocs voisins dans l'image précédente.

Le standard MPEG (ISO 1990, Gall 1991, Bhaskarau et Koustantinides 1995) (Motion Picture Expert Group) fonctionne selon ce principe. On distingue ainsi entre trois types d'images (figure 1.4):

1. **Images de type** I ces images sont des images de mise à jour et sont transmises intégralement et codées dans le domaine spatial par transformation en cosinus discrète (TCD).

2. **Images de type** P ces images sont codées par compensation du mouvement par rapport aux images I. On calcule ainsi pour chaque bloc (16×16) de

l'image P un vecteur de correspondance avec un bloc de l'image I. Ce vecteur est transmis avec l'erreur de reconstruction (la différence entre la prédiction et l'image originale P) codée avec une méthode de transformation TCD.

3. **Images de type** B ces images sont reconstruites de façon bidirectionnelle par une combinaison linéaire des images I et P. On calcule ainsi pour chaque bloc (16×16) de l'image B deux vecteurs de déplacement un avec l'image I et l'autre avec l'image P. L'image sera prédite comme la moyenne des deux images reconstruites en se basant sur ces deux vecteurs. On transmet les deux vecteurs et l'erreur de reconstruction codée de la même manière que celle des images de type P.

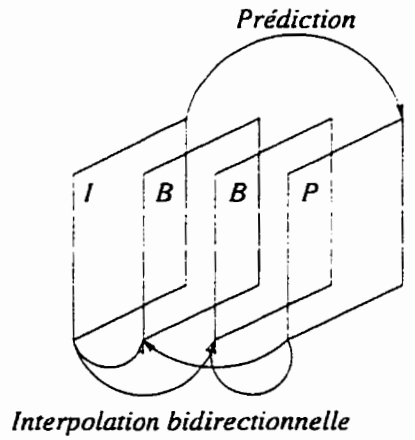


Figure 1.4 : La reconstruction dans le standard MPEG

D'autre travaux ont raffiné la méthode de compensation de mouvement par bloc en effectuant une classification des blocs de l'image et en les codant de manière différente selon leur classe (Wiegand, Lightstone, Campbell et Mitra 1995, Huang, Chen, Wang,

Hsieh et Hsieh 1994). La taille des blocs reste, quand même, fixe ce qui cause des erreurs gênantes à cause de l'hypothèse du mouvement uniforme dans chacun des blocs. En effet, attribuer un seul vecteur à un bloc revient à supposer que la portion de la scène projetée en ce bloc subit un mouvement rigide translationnel fronto-parallèle (divergence nulle). De plus, le fait de supposer que chaque bloc provient d'un seul objet dans la scène produit des erreurs aux niveaux des transitions entre les objets. Ceci oblige à utiliser des blocs de petite taille ce qui a pour effet de diminuer le taux de compression et de rendre le calcul des vecteurs de déplacement plus sensible au bruit. Certains travaux ont résolu ce problème en utilisant des blocs de taille variable (Zafar, Zhang et Jabbari 1993, Hang, Schilling et Puri 1987, Dufaux et Moscheni 1995, Sullivan et Baker 1991).

Le plus grand désavantage de la compensation de mouvement par blocs de taille variable est qu'elle est très coûteuse en temps de calcul. De plus, ces méthodes sont limitées à des blocs carrés et ne permettent pas la fusion des blocs adjacents par opposition aux techniques par régions polygonales (section 1.3.2.1). Par exemple, même si les vecteurs de déplacement des blocs b_{23} et b_{24} de la figure 1.5 se ressemblent, la fusion de ces derniers n'est pas permise.

Enfin, d'autres chercheurs se proposent d'améliorer la qualité des images obtenues en utilisant des meilleures descriptions du mouvement par bloc (Moccagatta, Moscheni, Schütz et Dufaux 1994, Orchard 1993, Fuldseth et Ramstad 1995).

Pour une description détaillée de toutes ces approches veuillez vous référer à la

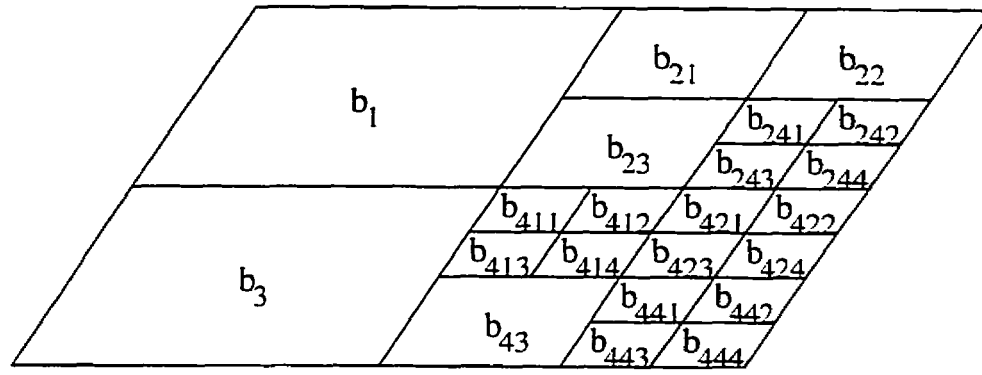


Figure 1.5 : Division d'une image en blocs de taille variable

section A.2.

L'effet de bloc, principale forme de dégradation Le plus grand problème avec les méthodes de compensation par bloc vient de l'effet de bloc (Li et al. 1994). Il provient de l'attribution, indépendante d'un bloc à ses voisins, du vecteur de déplacement. Ceci cause des discontinuités dans le champ de déplacement aux frontières des blocs. Plus on augmente le taux de compression plus la démarcation entre les blocs va être visible. La raison principale de ceci est le choix arbitraire de blocs rectangulaires, dans la mesure où ce type de régions augmente l'effet de bloc à cause de la sensibilité du système visuel humain aux détails horizontaux et verticaux. Les deux techniques les plus souvent utilisées pour réduire la visibilité de l'effet de blocs reposent sur le concept de blocs superposés (Ohta et Nogaki 1993, Young et Kingsbury 1993) et sur le post-traitement des images reçues au récepteur (Nakajima, Hori et Kanoh 1994). A la section A.2, nous présentons plusieurs approches basées sur ces deux techniques.

1.3.2 Compensation par région

Comme nous les avons déjà discutées, les techniques de compensation par bloc causent des erreurs gênantes aux bordures des blocs (l'effet de bloc). La raison principale de ceci est que les bordures des objets en mouvement ne coïncident pas avec les bordures des blocs. Il est évident que si nous segmentons l'image en régions ayant chacune un modèle différent de mouvement, la démarcation entre les différentes classes de mouvement sera placée aux occlusions alors que le système visuel humain est moins sensible aux détails à ces endroits. Ceci aura, certainement, pour effet d'augmenter la qualité des images perçues au récepteur (Li et al. 1994).

Plusieurs chercheurs (Bartolini, Cappellini, Mecocci et Vagheggi 1994, Carpentieri et Storer 1992, Carpentieri et Storer 1994, Leonardi et Chen 1994, Chan, Yu et Constantinides 1990, Lee 1995) ont généralisé les techniques par bloc en fusionnant plusieurs blocs pour former des régions polygonales. D'autres (Papadopoulos et Clarkson 1993, Nakaya et Harashima 1994, Chae, Kim et Park 1993, Dang, Mansouri et Konrad 1995, Wu, Benois-Pineau et Barba 1995, Salari et Lin 1995, Bouthemy et Francois 1993, Zheng et Blostein 1995) ont segmenté les images pour obtenir des régions de forme arbitraire dont ils modélisent le mouvement. Enfin, d'autres chercheurs (Wang et Lee 1994, Wang, Hsieh, Hu et Lee 1995, Altunbasak, Tekalp et Bozdagi 1995, Nieweglowski, Campbell et Haavisto 1993) ont adapté la grille des positions des pixels dans les images au contenu de ces dernières.

1.3.2.1 Compensation par régions polygonales

Dans ces techniques, plusieurs blocs sont réunis pour former des régions. La décision d'unir ou de diviser des régions se fait pour optimiser un critère. Les critères les plus souvent utilisés dans la littérature sont les suivants:

1. **Ressemblance des vecteurs de déplacement** la décision d'attribuer un bloc à une région se fait en se basant sur la différence entre le vecteur de déplacement du bloc et la moyenne des vecteurs de la région (Bartolini et al. 1994, Carpentieri et Storer 1992, Carpentieri et Storer 1994).

2. **Erreur de reconstruction** (EQM) on subdivise une région si l'erreur de reconstruction qui lui est associée est grande. Pour fusionner des régions, on se base sur le critère précédent (ressemblance des vecteurs de déplacement) (Leonardi et Chen 1994, Chan et al. 1990).

3. **Rapport taux/distorsion** les deux critères précédents effectuent la segmentation sans tenir compte de l'ajout d'information de segmentation à transmettre à chaque fois qu'on subdivise une région. Il peut être plus pertinent de se baser plutôt sur une fonction de coût qui tient compte de la qualité des images et de la compression obtenue (Lee 1995).

Pour une description détaillée de toutes ces approches, veuillez vous référer à la section A.3.1.

Ces approches sont des compromis sur les approches basées-bloc. La forme des régions est limitée à des unions de blocs et le mouvement de ces régions est représenté par un modèle translationnel (section 1.2.1). Elles ne font en définitive qu'essayer de diminuer l'effet de bloc.

1.3.2.2 Compensation par régions à modèle de mouvement

Une autre approche consiste à segmenter la séquence d'images et représenter le mouvement de chaque région par un des modèles présentés à la section 1.2.1. Il est à noter que les pixels déplaçés (x', y') correspondent en général à des sous-pixels et il faut donc effectuer une des interpolations décrites à la section 4.4 pour reconstruire les images.

Les différentes méthodes se basant sur ce concept différent par leur manière d'aborder le problème de la segmentation. En effet, il est possible de baser les modèles de mouvement sur des régions constantes (des blocs (Papadopoulos et Clarkson 1993) ou des triangles et des quadrilatères (Nakaya et Harashima 1994)) sauf que ceci cause les mêmes problèmes que les techniques par bloc. Les chercheurs optent, plutôt, pour une segmentation des images dans le domaine spatial[2] ou temporel[3]. La segmentation du mouvement nous permet de déterminer les objets qui bougent et nous fournit une bonne continuité temporelle (Chae et al. 1993). Bien sûr, cette segmentation souffre de la complexité et la sensibilité au bruit de l'estimation du mouvement. D'autre

[2]En se basant sur les intensités lumineuses des images
[3]En se basant sur les vecteurs de déplacement

part, la segmentation spatiale nous apporte une meilleure localisation mais tend à sur-segmenter les régions texturées (Dang et al. 1995, Wu et al. 1995, Salari et Lin 1995). Il est possible d'incorporer ces deux informations pour obtenir une segmentation spatio-temporelle. Enfin, il est possible de combiner la résolution du problème de la segmentation et de celui de l'estimation du mouvement. Il s'agit de déterminer le nombre de régions, l'allocation des pixels à ces régions et les paramètres de mouvement de chacune de ces régions pour minimiser une fonctionnelle. L'erreur de reconstruction est souvent le choix des chercheurs à cause de sa simplicité (Bouthemy et Francois 1993, Zheng et Blostein 1995). Ces méthodes sont décrites en détail à la section A.3.2.

1.3.2.3 Compensation par grilles adaptatives

Au lieu de représenter l'image par des pixels localisés sur une grille uniforme sur l'image, Wang et Lee (1994) proposent de se baser sur des noeuds ayant plus de densité aux discontinuités spatiales. Ils ont modélisé l'image comme des noeuds reliés par des ressorts ayant des constantes qui dépendent du contenu spatial de l'image au noeud.

Cette représentation peut être utilisée pour diviser l'image en tâches ayant chacun un mouvement différent. Plusieurs formes de tâches sont possibles. Des quadrilatères sont souvent utilisés (Wang et al. 1995). Le mouvement de chacun des quadrilatères est modélisé par un modèle bilinéaire à huit paramètres (équation 1.5). Dans la

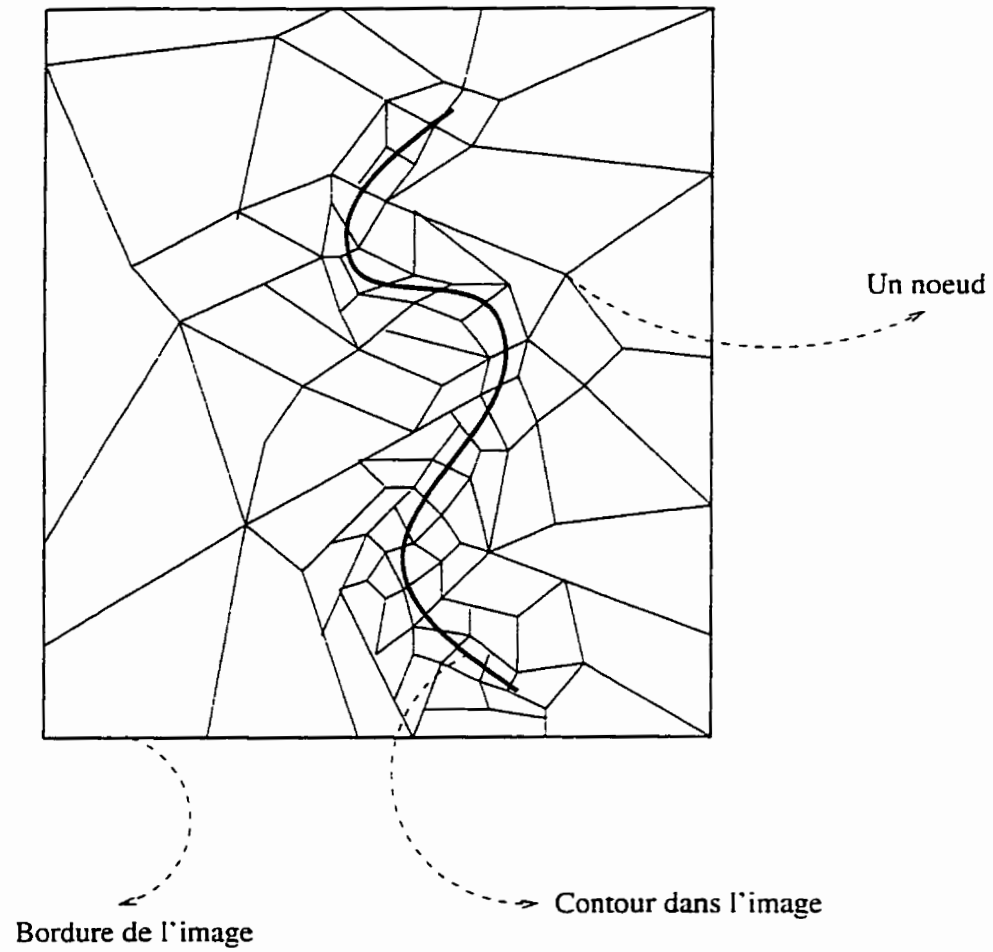


Figure 1.6 : Exemple d'une segmentation d'image en quadrilatères

figure 1.6. nous pouvons voir un exemple d'une telle segmentation en quadrilatères. Les vecteurs de mouvement des quatre noeuds délimitant le quadrilatère sont utilisés pour estimer les huit paramètres du modèle[4]. Il est aussi possible de diviser l'image en triangles dont le mouvement est représenté par un modèle affine à six paramètres (Altunbasak et al. 1995, Nieweglowski et al. 1993) (équation 1.6).

[4]En résolvant un système de huit équations (une par composante de vecteur) à huit inconnues

1.3.3 Compensation dense

Ces techniques n'imposent aucune restriction sur le mouvement de la scène. Un champ dense ($\vec{v}$) composé d'un vecteur par pixel doit donc être estimé et codé. Les tendances actuelles optent pour faire ceci de manière récursive afin de minimiser l'erreur de reconstruction DTD (Différence entre les Trames Déplacées) (Musmann, Pirsh et Grallert 1985). Cette erreur peut être exprimée selon l'équation:

$$DTD_t(x, y, \vec{v}) = I_t(x, y) - I_{t-1}(x - v_1, y - v_2) \tag{1.13}$$

En 1978, Netravali et Robbins (1979) ont publié un premier algorithme pour l'estimation du mouvement qui minimise l'erreur de reconstruction de façon récursive par descente du gradient selon les équations:

$$\vec{v}^{k+1} = \vec{v}^{k} - \frac{\epsilon}{2}\vec{\nabla}_{\vec{v}} DTD_t^2(x, y, \vec{v}^{k}) \tag{1.14}$$

$$\vec{\nabla}_{\vec{v}} DTD_t^2(x, y, \vec{v}) = 2DTD_t(x, y, \vec{v})\vec{\nabla} I_{t-1}(x - v_1, y - v_2) \tag{1.15}$$

où $\vec{\nabla}_{\vec{v}} f = (\frac{\partial f}{\partial v_1}, \frac{\partial f}{\partial v_1})$, ϵ est le gain d'adaptation et k est le numéro de l'itération. Les itérations peuvent être effectuées spatialement ou temporellement. Depuis, plusieurs autres algorithmes (Bergmann 1982, Cafforio et Rocca 1983, Bergmann 1984) ont été proposés pour adapter le gain ϵ au contenu spatial des images amenant à une augmentation de la vitesse de convergence.

Une version récente de ce genre d'algorithme est proposée par Huang et Mersereau

(1994). On décompose la différence entre les trames en deux parties: une due au mouvement, l'autre non. On estime deux déplacements par pixel un avec l'image précédente et l'autre avec l'image successive ce qui permet la reconstruction dans les deux sens de l'image courante.

Ces techniques récursives nous permettent d'estimer un champ dense de vecteurs au récepteur. Ce dernier utilise ce champ pour reconstruire les images avec une des méthodes d'interpolation présentées à la section 4.4. L'émetteur transmet donc uniquement l'erreur de reconstruction. Les différentes techniques pour transmettre cette erreur sont décrites à la section A.4.1.

Malheureusement, le champ dense obtenu en se basant sur ces techniques récursives est très sensible au bruit. Une alternative est de transmettre les accélérations à la place des vecteurs de déplacement (Yeh, Vetterli et Khansari 1995). Nous supposons que les pixels gardent le même déplacement dans le temps. Ceci peut être exprimé selon l'équation suivante:

$$\hat{\vec{v}}_{t+1}(x + v_{t_1}(x,y), y + v_{t_2}(x,y)) = \vec{v}_t(x,y) \tag{1.16}$$

où $\vec{v}_t(x,y) = (v_{t_1}(x,y), v_{t_2}(x,y))$. Nous transmettons $\vec{v}_{t+1} - \hat{\vec{v}}_{t+1}$. Le champ $\hat{\vec{v}}_{t+1}$, calculé à partir de l'équation 1.16, doit être interpolé pour correspondre à un vecteur par pixel au lieu d'un vecteur par sous-pixel.

Une autre altérnative est d'estimer et transmettre le mouvement uniquement aux contours (Murayama, Miyauchi et Shirota 1995). Les images sont reconstruites au

récepteur par interpolation à partir de ces contours. De plus, des contours spatiaux de mise à jour doivent être transmis de temps en temps. Un codage en chaine est utilisé à cette fin.

D'autre part, nous pouvons ne pas transmettre les vecteurs de mouvement (Baaziz et Labit 1994). Nous utilisons alors un algorithme symétrique où l'estimation du mouvement se fait à l'émetteur comme au récepteur. Nous utilisons le mouvement estimé au temps t pour effectuer la reconstruction au temps $t+1$ et les erreurs de reconstruction sont transmises.

Enfin, des approches en multirésolution ont été proposées pour estimer et coder le mouvement (Moulin et Loui 1993, Krishnamurthy, Moulin et Woods 1995, Haddadi et Kuo 1994*a*, Haddadi et Kuo 1994*b*). Ces approches sont expliquées en détail à la section A.4.2.

1.4 Discussion

La compensation de mouvement est une méthode prédictive de codage des séquences d'images. La prédiction est effectuée temporellement. Plus les vecteurs de mouvement $\vec{v}(x,y) = (v_1(x,y), v_2(x,y))$ sont de bonne qualité, moins la distorsion dans les images prédites au récepteur sera grande.

Les méthodes utilisées pour la compensation de mouvement imposent beaucoup de restrictions sur le flux optique comme nous l'avons vu à la section 1.3. Ceci a pour conséquence de diminuer la qualité des images prédites. Les chercheurs remédient à ce

problème en transmettant l'erreur de reconstruction. Plus les restrictions sur le flux sont grandes, plus l'erreur de reconstruction est volumineuse à transmettre. De plus, cette erreur n'est, en général, pas très corrélée spatialement et n'a, par conséquent, pas beaucoup de potentiel pour le codage (Ohta et Nogaki 1993). Il faut donc choisir un compromis entre la quantité d'information à transmettre pour le mouvement et pour l'erreur de reconstruction.

Les restrictions posées dans les techniques par bloc sont trop fortes. En effet, même si on se basait sur un modèle plus précis que celui à deux paramètres (modèle translationnel), la qualité des images prédites reste quand même inacceptable. La raison principale de ceci est l'estimation des paramètres du modèle dans des régions fixes (des blocs) qui n'appartiennent pas nécessairement à un seul objet en mouvement alors que tous les modèles proposés supposent un seul objet en mouvement rigide (section 1.2.1). Ceci se traduit en une erreur de reconstruction élevée, qu'on ne peut pas compresser pour des applications à bas débit tout en gardant une qualité acceptable des images perçues au récepteur.

Pour se défaire de la restriction des régions fixes dans les techniques par bloc, des stratégies par blocs de taille variable et par régions polygonales ont été proposées. Ces stratégies souffrent de leur grande complexité des calculs à cause de l'exigence d'estimer les paramètres de mouvement plusieurs fois. Ce problème peut être résolu en faisant une classification des blocs et en adaptant la taille de ces derniers à cette classification. Un entête doit être transmis pour indiquer la classe des blocs.

Des régions de forme arbitraire sont aussi souvent utilisées. On essaie alors de détecter des régions à mouvement homogène et de modéliser le mouvement dans ces régions. Le plus grand problème commun à ces approches est la dualité entre la segmentation et l'estimation du mouvement. En effet, pour pouvoir calculer le mouvement, il faut savoir sur quelle région l'estimer et pour pouvoir déterminer la segmentation, il faut savoir quels pixels ont un mouvement homogène et donc, il faut connaître leur mouvement. D'autre part, l'information de segmentation est très coûteuse à transmettre (de l'ordre de 1bit/point du contour de la région (Leonardi et Chen 1994)). Ce problème peut être résolu en estimant la segmentation au récepteur, comme à l'émetteur, une image en retard (Bonnaud et Labit 1994, Salari et Lin 1995).

En se basant sur le flux optique dense pour faire la compensation, nous résolvons tous les problèmes dans les techniques par bloc et par région. Le problème est que ce champ contient le double de la quantité d'information dans les images. En effet, pour chaque pixel, nous devons coder l'information d'amplitude et de la phase du vecteur de mouvement correspondant à ce pixel. Le codage du flux peut alors nous paraître laborieux. Les techniques de compensation dense récursive ont été alors proposées pour estimer ce champ dense au récepteur. Ces techniques souffrent des trois problèmes suivants (Li et al. 1994, Dufaux et Moscheni 1995):

1. la complexité des calculs;

2. la convergence à des minimums locaux;

3. le mauvais calcul des grands déplacements.

En effet, la qualité du flux obtenu avec ces techniques est même inférieure à un champ de déplacement translationnel par bloc de 8×8 (Li et al. 1994, Dufaux et Moscheni 1995).

Dans ce mémoire, nous proposons un système de compression des séquences d'images se basant sur le flux optique dense calculé auparavant. Le fait de se baser sur le mouvement dense pour faire la compensation de mouvement nous évite de transmettre l'erreur de reconstruction et nous permet par conséquent de coder les séquences à très bas débit. Une accumulation d'erreur est par contre causée par les erreurs dans le calcul du flux optique et dans la reconstruction des images par interpolation. Nous pallions à ce problème en transmettant une image de mise à jour quand cette erreur devient trop grande.

Dans le chapitre suivant, nous présentons la structure générale de notre système de codage.

Chapitre 2

Présentation générale du système

Notre objectif est de développer un système de codage de séquences d'images permettant la transmission de ces images dans des canaux déjà existants et qui sont dédiés pour la transmission de la voix à bas débit.

Dans le chapitre précédent, nous avons présenté le codage des séquences d'images par compensation de mouvement comme une méthode pour exploiter les redondances temporelles dans les séquences d'images. Hélas, le mouvement dans ces séquences a été mal modélisé ce qui causait des erreurs de reconstruction élevées, qu'on ne peut pas compresser pour des applications à bas débit tout en gardant une qualité acceptable des images perçues au récepteur. Nous proposons une meilleure description du mouvement basée sur le mouvement dense estimé par le flux optique.

2.1 Description du mouvement basée sur le flux optique

Le flux optique a été rarement exploité pour décrire le mouvement en compensation de mouvement. Ceci est dû aux deux raisons suivantes:

1. le calcul du flux optique est très couteux:
2. le codage du flux optique peut paraître laborieux car ce dernier contient le double de la quantité d'information dans les images (un vecteur par pixel).

Pour répondre au premier problème, plusieurs algorithmes ont été proposés pour calculer le flux optique en temps réel. En effet, la complexité des calculs dans les techniques traditionnelles d'estimation du mouvement par appariement augmente quadratiquement en augmentant la taille de l'espace de recherche de correspondance. Nous pouvons voir à la figure 2.1 que quand la vitesse d'un pixel double, l'air de l'espace de recherche de sa nouvelle position quadruple car cet espace constitue un cercle centré autour du pixel avec comme rayon la vitesse maximale que nous désirons détecter. La vitesse ($\vec{v}$) est reliée à la distance parcourue (Δd) et le temps (Δt) par la relation

$$\vec{v} = \frac{\Delta d}{\Delta t} \tag{2.1}$$

Dans les approches classiques d'estimation du mouvement par appariement, nous fixons le délai à une valeur δt correspondant à l'image précédente et nous cherchons la

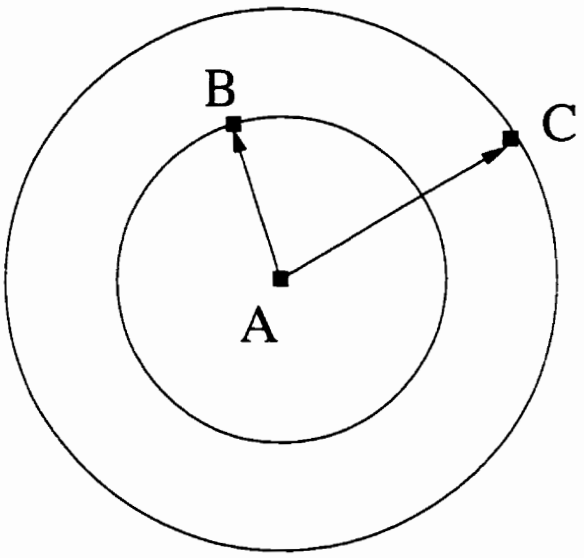


Figure 2.1 : Lorsque le le déplacement maximal des pixels augmente linéairement. l'espace de recherche augmente quadratiquement

correspondance spatiallement ($\vec{v} = \frac{\Delta d}{\delta t}$) ce qui résulte en un algorithme de recherche qui est quadratique selon les vitesses présentes dans la scène. Camus (1994) propose de rendre l'espace de recherche de correpondance linéaire au lieu de bidimentionnelle. Il réalise ceci en fixant la taille de l'espace de recherche spatial à δd et variant Δt en réalisant les appariements dans le temps. Il fixe $\delta d = 1$ et limite la recherche dans le temps aux S images précédentes $1 \leq \Delta t \leq S$. La plus petite vitesse détectable est alors de $1/S$ pixel/image. Le gain de performance apporté permet de calculer des champs de mouvement dense en temps réel pour des petites images (64 × 64 pixels).

D'autre part, Benoit et Ferrie (1996) se basent sur un schéma qui ressemble aux méthodes d'Anandan (Anandan 1987, Anandan 1989) et de Singh (1992). Ils découplent le calcul de la mesure de distance et la régularisation du champ. De plus, ils optimisent le nombre de fois qu'ils effectuent chacune de ces deux opérations. Une

implantation en logiciel leur permet de calculer des champs de mouvement dense à la fréquence de 4 images/sec ce qui promet un fonctionnement en matériel en temps réel.

Enfin, la plupart des séquences d'images contiennent des objets en mouvement et d'autres qui sont statiques. En détectant les régions statiques, nous pouvons éviter le calcul du flux optique à ces régions d'où une grande diminution du temps de calcul du flux optique (section 4.3.1).

Dans ce mémoire, nous résolvons le deuxième problème en exploitant les redondances spatiales et temporelles contenues dans le flux optique. Les redondances spatiales proviennent du fait que, dans des situations non-accidentielles, deux pixels adjacents ont souvent des vecteurs de mouvement semblables. D'autre part, les redondances temporelles proviennent de la continuité du mouvement des pixels dans le temps.

Notre système suppose l'existance de plusieurs sources de mouvement. Chaque source génére un type particulier de mouvement selon le modèle probabiliste de cette source. Le modèle des sources est appris avec un réseau de neurones à apprentissage Hebbien et est adapté dans le temps pour décrire de manière optimale le mouvement présent dans la scène. Le modèle appris au temps t est utilisé pour le codage de l'image au temps $t+1$. Ceci nous permet d'effectuer l'adaptation du modèle en même temps au récepteur et à l'émetteur et par conséquent d'éviter la transmission des modèles appris. Malgrè que notre modèle est à deux paramètres, il décrit le mouvement dense

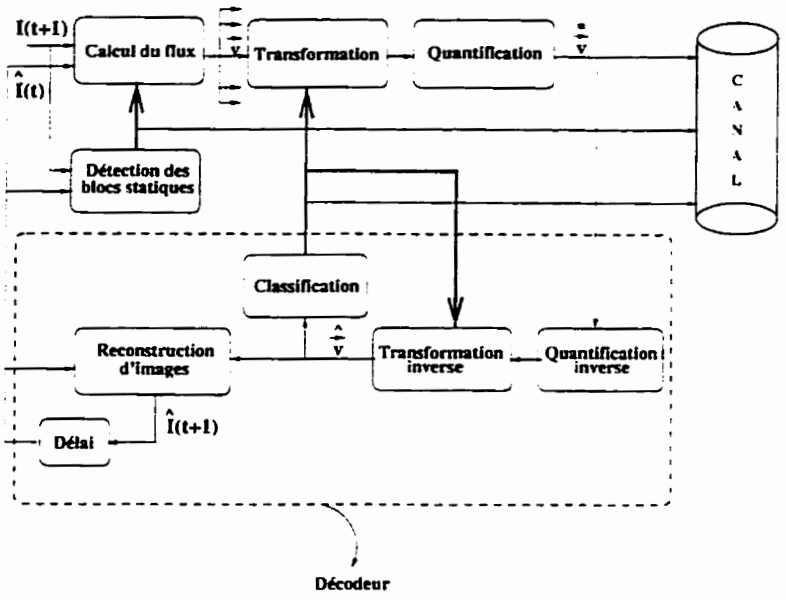


Figure 2.2 : Structure générale du système de compression vidéo

présent dans les images, par opposition au modèle translationnel à deux paramètres expliqué à la section 1.2.1. Cette meilleure description du mouvement nous évite la transmission de l'erreur de reconstruction et nous permet, par conséquent, de compresser les images pour des applications à bas débit.

La structure générale du système est présentée à la figure 2.2. Pour coder l'image $I(t+1)$, nous calculons, aux régions dynamiques[1], le flux optique entre cette image et l'image reconstruite à l'instant précédent $\hat{I}(t)$.

Ce flux est divisé en blocs ($\vec{v}$) et chacun de ces blocs est associé à une source de mouvement. Les deux paramètres décrivant le bloc dans le modèle de sa source sont évalués. La classification et les paramètres de chacun des blocs dynamiques sont quantifiés et transmis au récepteur. Ce dernier restitue le mouvement dense à partir

[1]Régions qui subissent un mouvement, par opposé aux régions statiques

du modèle et utilise ce mouvement pour reconstruire l'image $\hat{I}(t+1)$. Enfin, le réseau de neurones à apprentissage Hebbien est utilisé, à l'émetteur et au récepteur, pour adapter le modèle de mouvement de chacune des sources. Ces modèles adaptés seront utilisés pour coder le flux entre l'image $\hat{I}(t+1)$ et $I(t+2)$ pour reconstruire l'image $\hat{I}(t+2)$ au récepteur.

Comme nous allons le voir aux chapitres subséquents, le calcul du flux optique et la reconstruction des images à partir de ce dernier causent des erreurs inhérentes au système et qui s'accumulent dans le temps. Nous pallions à ce problème en transmettant, à chaque fois que l'erreur devient trop grande, des images de mise à jour codées spatialement.

Le diagramme fonctionnel du logiciel de simulation développé ainsi qu'un guide d'utilisation sont décrits à l'annexe B.

Le modèle de mouvement utilisé est décrit en détail dans le chapitre 3.

Chapitre 3

Représentation du mouvement

Comme nous l'avons mentionné auparavant, nous proposons de baser la compensation de mouvement sur le champ de mouvement apparent (flux optique) estimé. Pour compresser la quantité d'information contenue dans ce champ, un modèle de ce dernier sera développé afin de n'avoir à transmettre que les paramètres du modèle pour reconstruire une estimation du champ. Dans ce chapitre, nous présenterons le modèle de mouvement et décrirons plusieurs méthodes pour estimer les paramètres de ce modèle en temps réel.

3.1 Transformation du flux en composantes principales

Pour faciliter la compression du mouvement et la rendre parallélisable, le domaine de l'image est divisé en blocs. On se propose tout d'abord de décorréler par transformation le mouvement de chacun des blocs ($\vec{v}(x,y) = a(x,y)e^{j\theta(x,y)}$). Ceci permettra de mieux le compresser comme expliqué à la section 4.3.2.2.

Parmi les transformations linéaires les plus avantageuses sont les transformations orthonormales, car elles conservent la puissance du signal et leur transformée inverse peut être facilement obtenue par transposition.

Soit $x = [x(1), x(2), \ldots, x(N)]$ le vecteur de données, T la matrice de transformation et $y = [y(1), y(2), \ldots, y(N)]$ le vecteur de données dans l'espace transformé. Notons par $T(i)$ la $i^{\text{ème}}$ ligne de la matrice T. En d'autres mots, $T(i)$ est le $i^{\text{ème}}$ vecteur de base de la transformation. Nous avons donc:

$$y(i) = T(i)\, x^t \tag{3.1}$$

$$y^t = T\, x^t \tag{3.2}$$

Puisque la matrice de transformation est orthonormale, la puissance du signal est conservée comme nous pouvons le voir dans l'équation suivante:

$$\|y\|^2 = yy^t = xT^tTx^t = \|x\|^2 \tag{3.3}$$

La compression des données consiste à réduire la dimension de l'espace transformé de N à M, avec $M < N$. Le vecteur reconstruit $\hat{x}$ peut être calculé par $\hat{x} = \sum_{i=1}^{M} y(i)T(i)$. L'erreur introduite en négligeant $N - M$ termes est donnée par la relation:

$$\Delta x = x - \hat{x} = \sum_{i=M+1}^{N} y(i)T(i) \tag{3.4}$$

L'erreur quadratique moyenne ϵ sera, par conséquent:

$$\epsilon = E[\Delta x \Delta x^t] = \sum_{i=M+1}^{N} \sum_{j=M+1}^{N} E[y(i)T(i)T(j)^t y(j)^t] = \sum_{i=M+1}^{N} E[y(i)^2] \tag{3.5}$$

Pour trouver la matrice de transformation qui minimise cette erreur tout en gardant les bases orthonormales, nous pouvons utiliser la technique des mutiplicateurs de Lagrange et maximiser pour la $i^{\text{ème}}$ composante:

$$\begin{aligned} E[y(i)^2] \quad - \quad & \lambda_1[T(i)T(i)^t - 1] - \lambda_2 T(i)T(j)^t = \\ & T(i)C_x T(i)^t - \lambda_1[T(i)T(i)^t - 1] - \lambda_2 T(i)T(j)^t \quad \forall j \end{aligned} \tag{3.6}$$

où C_x est la matrice de covariance de l'entrée x et λ_1, λ_2 sont des mutiplicateurs de Lagrange. Dans la relation précédente, le terme que multiplie λ_1 correspond à la contrainte de normalisation des vecteurs de base et le terme que multiplie λ_2 correspond à la contrainte d'orthogonalité des vecteurs de base. Le signe $(-)$ devant le deuxième et le troisième termes implique la minimisation de ces derniers. Le maximum de cette

relation se produit quand sa dérivée par rapport à $T(i)$ est nulle, et donc:

$$C_x T(i)^t - \lambda_1 T(i)^t - \lambda_2 T(j)^t = 0 \quad \forall j \tag{3.7}$$

En multipliant 3.7 à gauche par $T(j)$, nous trouvons:

$$T(j)C_x T(i)^t - \lambda_1 T(j)T(i)^t - \lambda_2 T(j)T(j)^t = 0 \quad \forall j \tag{3.8}$$

Le premier terme de cette relation est égal à la covariance entre $y(i)$ et $y(j)$[1] et est donc nul par la définition même de la matrice de transformation. De plus, le deuxième terme est nul à cause de l'orthogonalité des bases. Enfin, $T(j)T(j)^t = 1$ et donc $\lambda_2 = 0$. En d'autre mots, en maximisant les deux premiers termes de la relation 3.6, le troisième est automatiquement maximisé.

La relation 3.7 peut donc s'écrire sous la forme:

$$(C_x - \lambda_1 I_N)T(i)^t = 0 \tag{3.9}$$

où I_N est la matrice identité de taille $N \times N$.

La relation 3.9 montre que la matrice de transformation qui minimise l'erreur quadratique moyenne de reconstruction est celle ayant ses bases identiques aux vecteurs propres de la matrice de covariance du signal d'entrée. Sachant que $E[y(i)^2] =$

[1] $T(j)C_x T(i)^t = E[T(j)x^t x T(i)^t] = E[y(j)y(i)^t]$

$T(i)C_xT(i)^t = T(i)\lambda_1 T(i)^t = \lambda_1$, il suffit de classer les vecteurs propres de C_x en ordre décroissant de leur valeur propre pour que les vecteurs de base de la transformation soient classées en ordre décroissant de leur énergie.

Cette transformation s'appelle la transformation en composantes principales TCP ou la transformée de Karhunen-Loève TKL (Jolliffe 1986). Nous allons appeler $T(i)$ la $i^{\text{ème}}$ composante principale de la transformation et $y(i)$ le $i^{\text{ème}}$ coefficient principal du signal transformé. $T(1)$ sera désigné par le terme composante principale et $y(1)$ par le terme coefficient principal.

Nous pouvons noter à l'équation 3.6 que la TCP est celle qui maximise, à tour de rôle, l'énergie contenue dans le $i^{\text{ème}}$ coefficient principal ($E[y(i)^2]$) tout en gardant les $i-1$ coefficients avant lui intact. En d'autres mots, elle classe les coefficients principaux en ordre décroissant dénergie. Par conséquent, la TCP d'un signal complexe (comme le champ de mouvement que nous avons à transformer) minimise l'erreur quadratique moyenne de reconstruction de l'amplitude de ce signal et ne tient pas compte de la phase de ce dernier. L'information de phase du mouvement étant très importante pour la reconstruction des images, nous devons effectuer la transformation indépendemment pour l'amplitude $a(x, y)$ et pour la phase $\theta(x, y)$. En d'autres mots, nous avons deux matrices de transformation à apprendre: T^a pour l'amplitude et T^θ pour la phase.

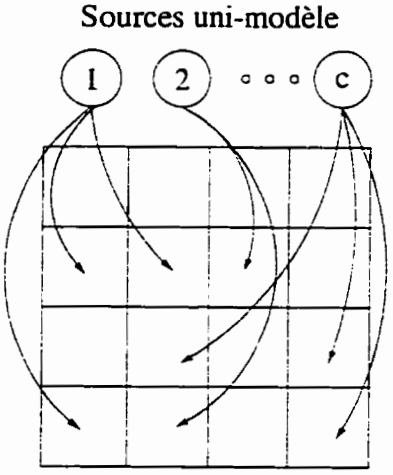


Figure 3.1 : Classification du mouvement des blocs en régions stationnaires

3.2 Modèle à source composée

Pour pouvoir apprendre la TCP du mouvement et l'adapter en temps réel, il faut que ce dernier soit stationnaire. Or ceci n'est en général vrai que localement à cause de la présence de plusieurs objets en mouvement dans les séquences d'images. Par conséquent, le mouvement dense doit être segmenté en plusieurs classes ayant chacune un mouvement stationnaire généré par une source indépendante des autres sources comme illustré à la figure 3.1.

Le champ de mouvement dense étant multi-modèles, il peut donc être généré par c sources dont chacune est uni-modèle. Dans la figure 3.2, la distribution du mouvement généré par une source est représentée par la composante principale de l'amplitude et de la phase du mouvement des blocs qui y sont attribués. Soit T_i^a et T_i^θ la matrice de transformation de l'amplitude et de la phase respectivement de la $i^{\text{ème}}$ source. Soit un bloc de mouvement $\{\ \vec{v}(x,y) = a(x,y)e^{j\theta(x,y)}\ x,y \in [1,\sqrt{N}]\ \}$

de taille $\sqrt{N} \times \sqrt{N}$. Nous allons représenter $a(x, y)$ et $\theta(x, y)$ par les vecteurs (de taille N) $\tilde{a}$ et $\tilde{\theta}$ respectivement qui correspondent aux lignes des blocs plaçées l'une à la suite de l'autre comme l'exprime les équations suivantes:

$$\tilde{a}\,(u) = a(\lfloor \frac{u-1}{\sqrt{N}} \rfloor + 1, u - \lfloor \frac{u-1}{\sqrt{N}} \rfloor \times \sqrt{N}) \tag{3.10}$$

$$\tilde{\theta}\,(u) = \theta(\lfloor \frac{u-1}{\sqrt{N}} \rfloor, u - \lfloor \frac{u-1}{\sqrt{N}} \rfloor \times \sqrt{N}) \tag{3.11}$$

Ce bloc de mouvement peut être exprimé dans le domaine transformé de la $i^{\text{ème}}$ source par:

$$\tilde{a} = \sum_{j=1}^{N} y_i^a(j)\ T_i^a(j) \tag{3.12}$$

$$\tilde{\theta} = \sum_{j=1}^{N} y_i^\theta(j)\ T_i^\theta(j) \tag{3.13}$$

$y_i^a(j)$ et $y_i^a(j)$ correspondent donc au $j^{\text{ème}}$ coefficient de l'amplitude et de la phase respectivement du bloc pour la $i^{\text{ème}}$ source. La probabilité de générer ce bloc peut donc être décrite selon l'équation:

$$P(a, \theta) = \sum_{i=1}^{c} P(i) P(a, \theta | i) \tag{3.14}$$

Chaque instance du modèle peut être complètement décrite par la classification des blocs (c^a et c^θ) et les coefficients principaux de l'amplitude ($y_{c^a}^a(i)\ \forall i$) et de la phase ($y_{c^\theta}^\theta(i)\ \forall i$) de chacun de ces blocs. La compression est effectuée en ne conservant

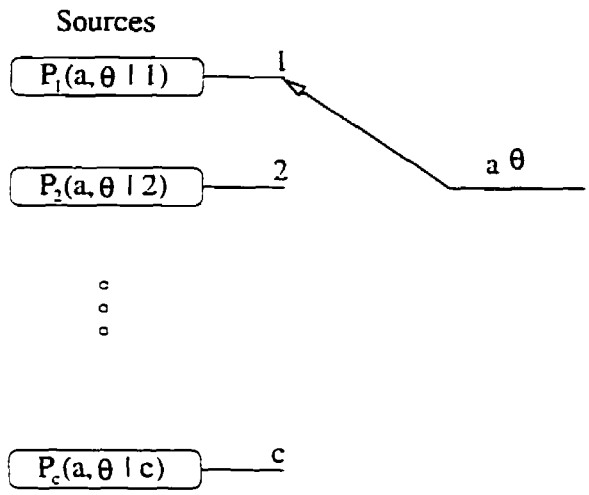


Figure 3.2 : Décomposition du mouvement en sources uni-modèles

qu'une partie des coefficients principaux ($y^a_{c^a}(i)$ et$y^\theta_{c^\theta}(i) \quad i < N$).

Malgré le fait que la TCP minimise l'erreur quadratique moyenne de reconstruction, elle a été rarement employée à cause de son implantation complexe. En effet, pour chaque nouveau signal à transformer, une matrice de covariance doit être évaluée et les vecteurs propres de cette dernière calculés et classés en ordre décroissant de leurs valeurs propres.

Dans la section suivante, nous allons présenter une autre alternative pour le calcul des matrices de transformation en composantes principales qui nous permet d'effectuer ce calcul itérativement et en temps réel.

3.3 Apprentissage des composantes principales par réseau de neurones

Comme mentionné auparavant, le calcul des matrices de transformation en composantes principales est très long et coûteux. En effet, le calcul de la matrice de covariance et le calcul des vecteurs propres sont de l'ordre de $O(N)$ et $O(M\sqrt{N})$ respectivement, où M correspond au nombre de coefficients à garder par bloc de $\sqrt{N} \times \sqrt{N}$ (Rao et Yip 1990).

Les réseaux de neurones linéaires à une couche qui utilisent la loi de Hebb pour adapter leurs poids de connexion ont montré leur capacité d'extraire les composantes principales d'une série de données avec moins de calcul. Dans ce qui suit, nous allons présenter deux réseaux, celui d' Oja (1982) qui permet d'extraire la composante principale avec un calcul de l'ordre de $O(\sqrt{N})$ et celui de Sanger (1989) qui permet de calculer les M composantes principales avec un calcul de l'ordre de $O(M\sqrt{N})$.

En 1949, Hebb (1949) a expliqué le fonctionnement d'une cellule de mémoire (un neurone) de la façon suivante:

- Si deux neurones situés de part et d'autre d'une synapse sont activés simultanément alors l'efficacité de cette synapse est augmentée.

- Si deux neurones situés de part et d'autre d'une synapse ne sont pas activés alors l'efficacité de cette synapse est diminuée.

Autrement dit, la mémoire devient un réflexe si un événement se produit souvent.

Par contre, on oublie les événements rares.

Le fonctionnement d'un neurone peut être modélisé comme un simple calcul linéaire (figure 3.3). La sortie y_i du neurone i est la somme des entrées $x^t = (x_1, x_2, \ldots, x_N)$ pondérées par les poids de connection $T_i^t = (T_{i1}, T_{i2}, \ldots, T_{iN})$. Ceci peut être décrit par l'équation:

$$y(i) = T(i)x^t = \sum_{j=1}^{N} T(i,j)x(j) \tag{3.15}$$

Une simple application de la loi de Hebb pour modifier les poids consiste en:

$$\Delta T(i,j) = \eta y(i)x(j) \tag{3.16}$$

où η est le taux d'apprentissage. Il est évident que cette règle d'apprentissage consiste à augmenter les poids T_i sans limite, ce qui n'est pas acceptable en réalité. Deux solutions sont possibles pour résoudre ce problème:

1. la **limitation des poids** consiste à les empêcher de dépasser un certain seuil. Elle est simple à implanter mais cause des effets d'écrêtage qui ne sont pas désirables;

2. pour obtenir un vecteur de connection de norme unitaire, nous pouvons opter pour la **normalisation des poids**. La contrainte est donc $\|T_i\|^2 = \sum_{j=1}^{N} T_{ij}^2 = 1$. Il est possible de prouver que la loi de Hebb est exprimée dans ce cas par

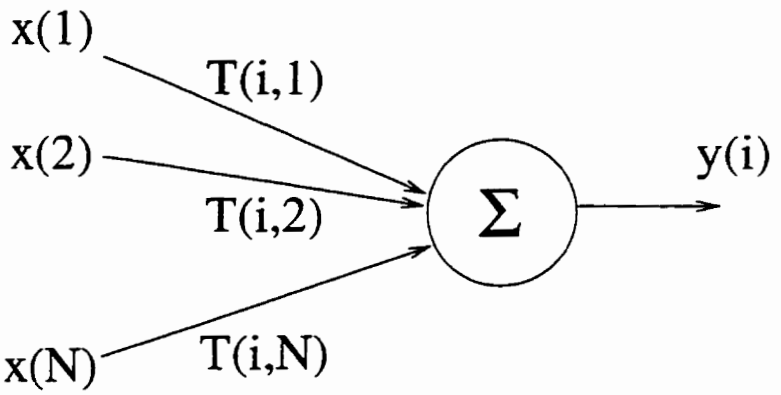


Figure 3.3 : Neurone linéaire simple

l'équation:

$$\Delta T(i,j) = \eta y(i)[x(j) - y(i)T(i,j)] \tag{3.17}$$

Oja (1982) a prouvé qu'en utilisant l'équation 3.17 pour modifier les poids du système de la figure 3.3. T_i converge vers la composante principale du signal d'entrée.

Plusieurs systèmes ont été proposés pour extraire les M composantes principales d'un signal. L'architecture générale de ces systèmes est montrée à la figure 3.4. Parmi les algorithmes généralisés de Hebb, nous avons utilisé celui de Sanger (1989) qui semblait donner des résultats à notre satisfaction. Le modèle de Sanger pour la modification des poids peut être exprimé en notations matricielles par:

$$\Delta T = \eta(y^t x - TI[y^t y]T) \tag{3.18}$$

où $TI[.]$ est une opération qui annule tous les éléments au-dessus de la diagonale d'une matrice (qui rend la matrice triangulaire inférieure). T converge vers la matrice

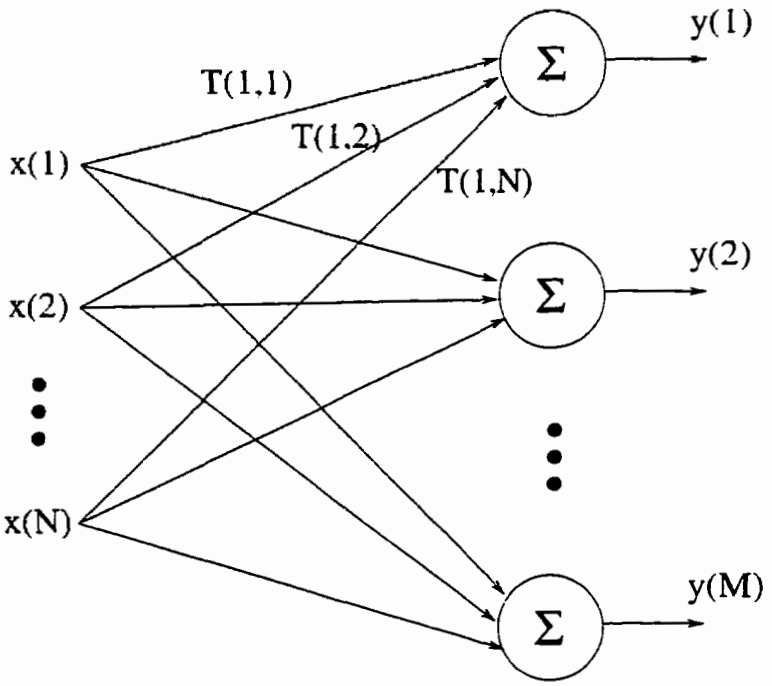


Figure 3.4 : Réseau linéaire pour extraire les M composantes principales

de transformation en composantes principales à condition que $\lim_{t\to\infty} \eta(t) = 0$ et $\sum_{t=0}^{\infty} \eta(t) = \infty$, où t est le numéro de l'itération.

Pour s'assurer de la convergence du système, nous avons initialisé le taux d'apprentissage à une grande valeur (0.2). Cette valeur est diminuée dans le temps selon la relation suivante[2]:

$$\eta(t) = \frac{1}{10t} \quad t \neq 0 \tag{3.19}$$

Aux figures 3.5 et 3.6, nous avons un exemple de l'application de cet algorithme pour la transformation d'un mouvement synthétique divergent et rotationnel respectivement. Ces figures montrent le mouvement reconstruit en utilisant les transforma-

[2]D'après la définition des séries harmoniques, cette relation vérifie les deux conditions de convergence ($\lim_{t\to\infty} \eta(t) = 0$ et $\sum_{t=0}^{\infty} \eta(t) = \infty$)

tions adaptées selon l'algorithme de Sanger (équation 3.18) comparée à la transformée en cosinus discrète (Rao et Yip 1990) qui est la transformée la plus utilisée dans la littérature. L'erreur d'angle Φ, comme décrite à la section 1.2, est une mesure d'erreur entre deux vecteurs de mouvement $\vec{u} = (u_1, u_2)$ et $\vec{v} = (v_1, v_2)$. Elle est exprimée par les équations:

$$\Phi = \cos^{-1} < \vec{v_N}, \vec{u_N} > \tag{3.20}$$

$$\vec{v_N} = \frac{(v_1, v_2, 1)^T}{\|\vec{v_N}\|}. \tag{3.21}$$

$$\vec{u_N} = \frac{(u_1, u_2, 1)^T}{\|\vec{u_N}\|} \tag{3.22}$$

où $< \ldots >$ désigne le produit scalaire.

L'erreur d'angle moyenne entre les flux reconstruits et le flux original est présentée dans les figures 3.5 et 3.6. Un gain de qualité de $\frac{9.39-1.99}{1.99} = 372\%$ et de $\frac{13.95-2.12}{2.12} = 558\%$ est atteint en conservant un coefficient de l'amplitude et de la phase de la TCP par rapport à la TCD pour le mouvement divergent et rotationnel respectivement.

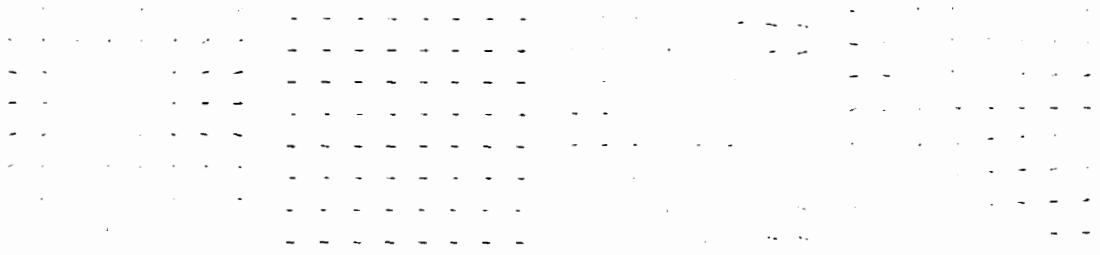
En conservant plus de coefficients pour la TCD, nous pouvons atteindre la même qualité que la TCP sauf que ceci se traduit par une compression moindre. Aux figures 3.5 et 3.6, nous remarquons qu'en concervant 6 et 17 coefficients de la TCD pour le mouvement divergent et rotationnel respectivement, nous obtenons la même qualité du mouvement reconstruit qu'en conservant 1 coefficient de la TCP. Ceci se traduit par un gain de compression de la TCP par rapport à la TCD de 6 : 1 et de

17 : 1 respectivement.

L'algorithme a été aussi appliqué pour transformer deux blocs réels de la séquence "tunnel"[3]. Le premier bloc (figure 3.7) correspond à un bloc du train en rotation et l'autre (figure 3.7) à un bloc situé à la frontière du train. Pour ce dernier, il n'est pas possible d'évaluer l'erreur d'angle car cette mesure n'est pas définie pour des vecteurs nuls, il est, par contre, facile d'évaluer visuellement la qualité du mouvement reconstruit.

Nous remarquons aux figures 3.5 à 3.7 que le réseau commence au début à osciller à cause du grand taux d'apprentissage. Par la suite, l'apprentissage devient de plus en plus raffiné avec la diminution du taux d'apprentissage dans le temps.

[3]Se réferer à la section 4.1 pour une desciption de la séquence

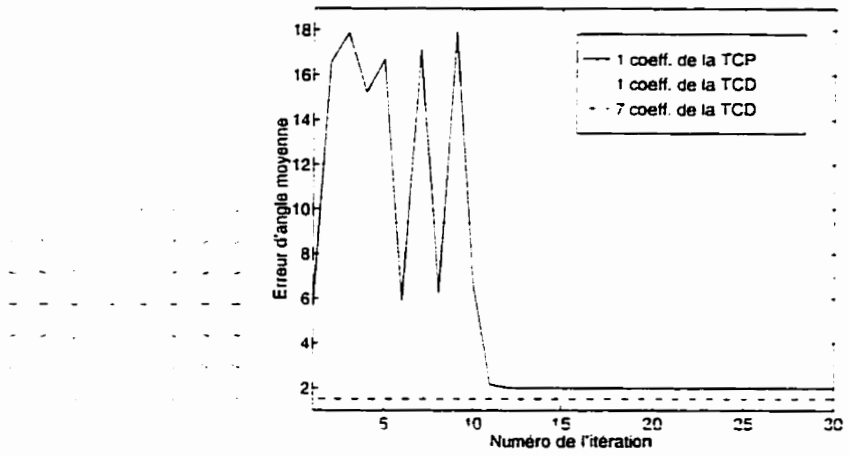


(a) Flux original

(b) Flux reconstruit en gardant un coefficient de la TCD

(c) Flux reconstruit en gardant 6 coefficients de la TCD

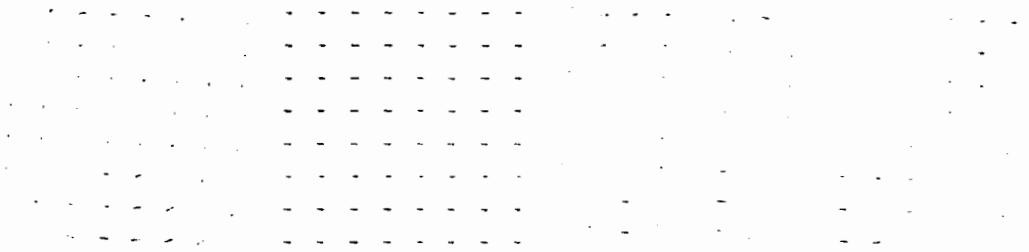
(d) Flux reconstruit en gardant un coefficient de la transformée adaptée après une itération



(e) Flux reconstruit en gardant un coefficient de la transformée adaptée après 14 itérations

(f) Erreur d'angle sur le flux reconstruit

Figure 3.5 : Exemple d'une compression d'un mouvement dense divergent avec la TCD et la TKL

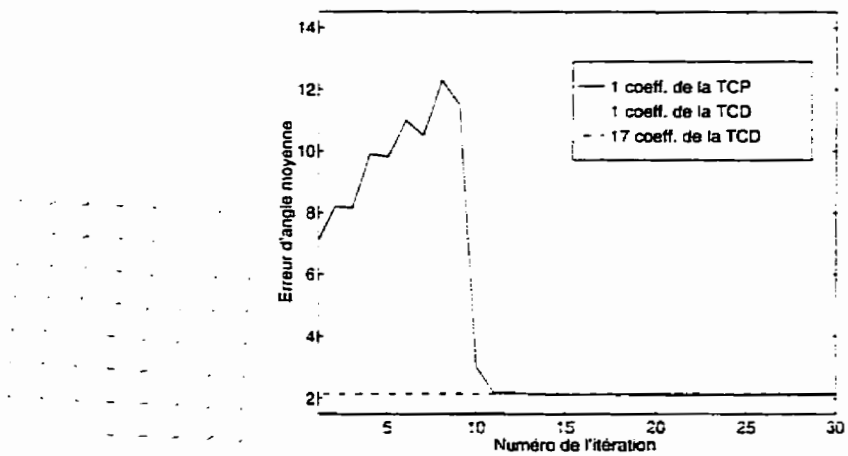


(a) Flux original

(b) Flux reconstruit en gardant un coefficient de la TCD

(c) Flux reconstruit en gardant 17 coefficients de la TCD

(d) Flux reconstruit en gardant un coefficient de la transformée adaptée après une itération



(e) Flux reconstruit en gardant un coefficient de la transformée adaptée après 15 itérations

(f) Erreur d'angle sur le flux reconstruit

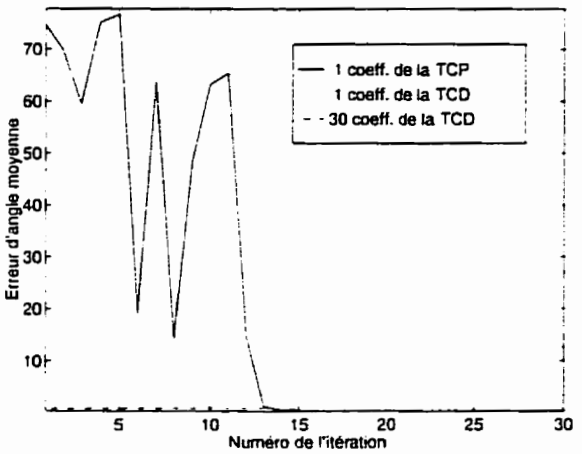
Figure 3.6 : Exemple d'une compression d'un mouvement dense rotationnel avec la TCD et la TKL

(a) Flux original

(b) Flux reconstruit en gardant un coefficient de la TCD

(c) Flux reconstruit en gardant 30 coefficients de la TCD

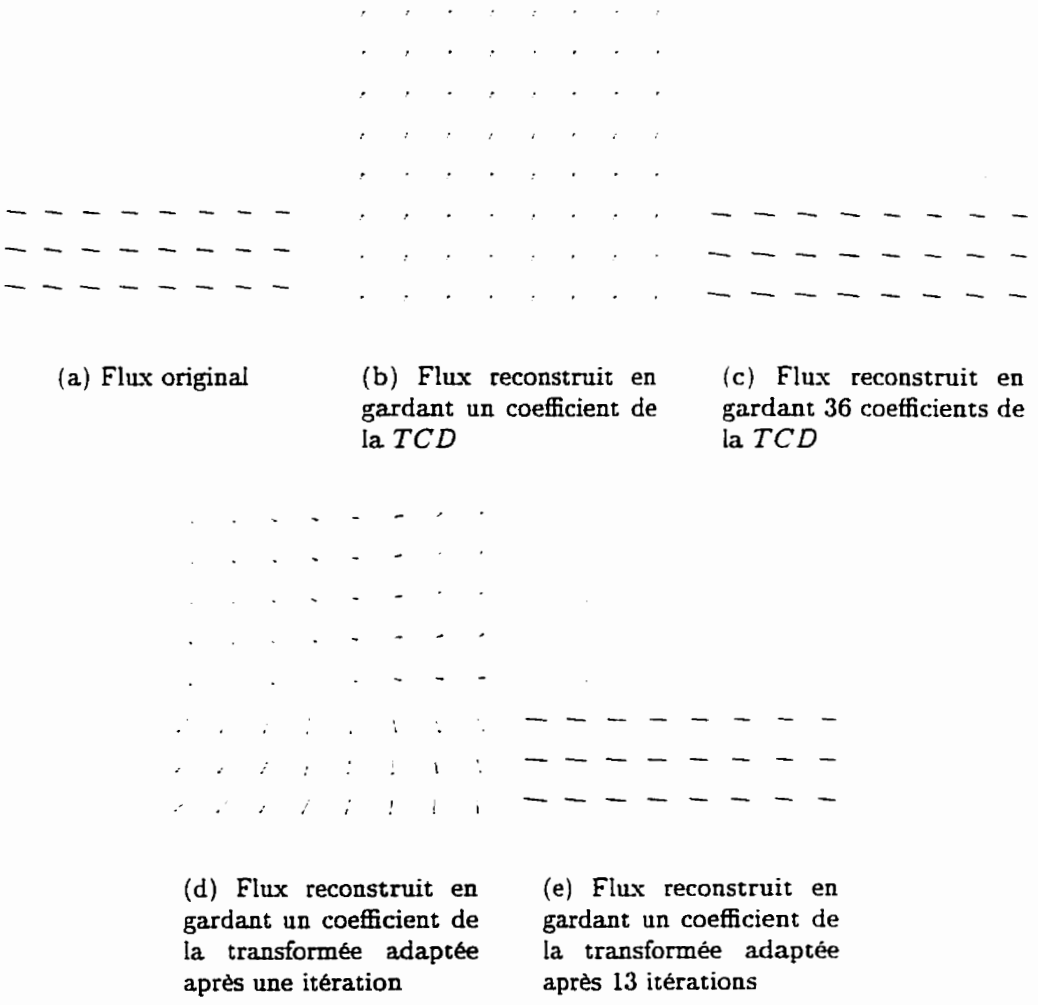
(d) Flux reconstruit en gardant un coefficient de la transformée adaptée après une itération



(e) Flux reconstruit en gardant un coefficient de la transformée adaptée après 20 itérations

(f) Erreur d'angle sur le flux reconstruit

Figure 3.7 : Exemple d'une compression d'un bloc de mouvement réel de la séquence "tunnel" avec la TCD et la TKL



(a) Flux original

(b) Flux reconstruit en gardant un coefficient de la TCD

(c) Flux reconstruit en gardant 36 coefficients de la TCD

(d) Flux reconstruit en gardant un coefficient de la transformée adaptée après une itération

(e) Flux reconstruit en gardant un coefficient de la transformée adaptée après 13 itérations

Figure 3.8 : Exemple d'une compression d'un bloc de mouvement réel situé à la frontière du train de la séquence "tunnel" avec la TCD et la TKL

Chapitre 4

Description de la méthode de compression vidéo

Dans ce chapitre, nous décrivons en détail le système de compression vidéo développé. Nous commençons par présenter les images qui ont été utilisées pour tester tous les aspects de ce système. Par la suite, nous décrivons chaque module du système. Enfin, nous discutons des détails d'implantation du système.

4.1 Présentation des séquences de test

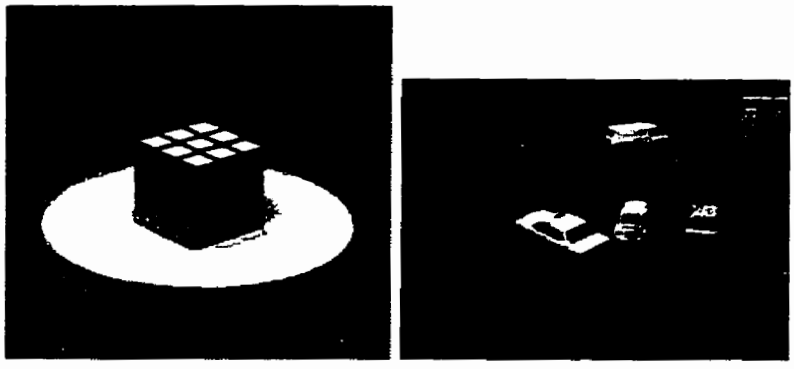
Les quatre séquences réelles de la figure 4.1 ont été utilisées pour tester les modules de notre système de compensation de mouvement. Le tableau 4.1 donne une description du mouvement dans chacune de ces séquences. Les trois premières séquences sont trop courtes pour nous permettre de bien étudier le comportement

du système en régime permanent. Pour obtenir des plus longues séquences. nous avons répété chacune de ces séquences plusieurs fois en va et vient. Par exemple. la séquence "taxi" utilisée dans les tests contient. en ordre chronologique. les images $01\cdots98\cdots01\cdots98\cdots01\cdots98\cdots01\cdots98\cdots3$. faisant une séquence de 70 images.

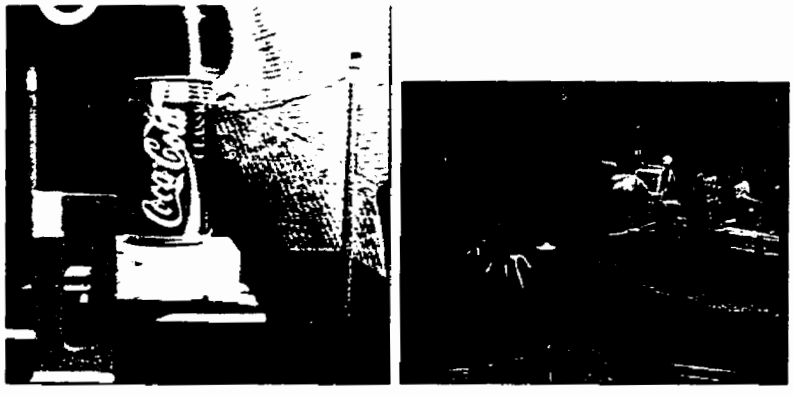
Les séquences "Rubik". "Taxi" et "Tunnel" contiennent beaucoup de régions statiques. En effet. le mouvement dans ces séquences est limité au cube. aux trois voitures et au train respectivement. Nous allons exploiter la présence de ces régions statiques en évitant la transmission des paramètres de mouvement dans ces régions. Par contre. le mouvement dans la séquence "Nasa" est un mouvement divergent de la caméra par rapport à une scène fixe. Ceci se traduit par des images globalement dynamiques. Dans le reste de ce chapitre. nous étudierons l'effet de ces caractéristiques sur la qualité des images reconstruites et les compressions obtenues.

Tableau 4.1 : Description des séquences types sur lesquelles tous les tests sont faits

Séquence	Nombre d'images	Description du mouvement
"Rubik"	20	Mouvement rotationnel tridimensionnel du cube et de la table tournante qui le supporte
"Taxi"	10	Mouvements de type frontal: rotationnel du taxi au milieu et translationnel des deux autres vehicules au bord gauche et droite de l'image
"Nasa"	35	Mouvement de translation tridimensionnelle globalement divergent
"Tunnel"	50	Mouvement rotationnel tridimensionnel du train



(a) Séquence "Rubik" (b) Séquence "Taxi"



(c) Séquence "Nasa" (d) Séquence "Tunnel"

Figure 4.1 : Première image de chacune des séquences types sur lesquelles tous les tests sont faits

4.2 Calcul du flux optique

Les méthodes les plus souvent utilisées pour calculer le flux optique sont les méthodes différentielles. Ces méthodes sont basées sur la contrainte du gradient comme expliqué à la section 1.2. Elles souffrent des deux problèmes suivants:

- **sensibilité au bruit** causée par le calcul des dérivées numériques des intensités lumineuses;

- **décalage entre l'emetteur et le récepteur**. En effet, le filtre numérique pour calculer les dérivées temporelles des intensités lumineuses exige un décalage entre l'émetteur et le récepteur.

Pour pallier à ces deux problèmes, les techniques d'appariement de caractéristiques qui ne se basent pas sur la contrainte du gradient peuvent être utilisées pour calculer le flux optique. Pour chaque pixel, des deux images successives, une primitive est évaluée. Ces primitives sont par la suite appariées entre les deux images. La primitive la plus souvent utilisée pour l'appariement est constituée d'un voisinage spatial (Anandan 1987, Anandan 1989, Singh 1992). Weng (Weng 1993, Weng, Ahuja et Huang 1992) suggère d'utiliser la phase de la transformation fenêtrée de Fourier car elle contient beaucoup d'information pour l'appariement tout en étant moins sensible au bruit. Sa technique a été utilisée dans notre système pour calculer le flux optique. Elle sera décrite dans la section suivante.

4.2.1 Calcul du flux optique basé sur la phase de la transformation fenêtrée de Fourier

Oppenheim et Lim (1981) ont étudié l'importance de la phase de la transformation de Fourier dans la représentation de l'information sur le signal considéré. Ils ont reconstruit des signaux originaux de test en faisant la transformation inverse de

Fourier à partir d'information partielle sur la transformée de ces signaux. L'expérience a montré qu'à partir de l'information de phase (en fixant l'amplitude partout à l'unité), il est possible de reconstruire une version identifiable du signal original. Par contre, en éliminant l'information de phase et en ne gardant que l'information d'amplitude, le signal reconstruit n'est pas discernable.

Ce résultat est exploité dans cette approche. En effet, la phase de la transformation fenêtrée de Fourier est utilisée comme primitive d'appariement. Le noyau de la transformation $h(x, y)$ est composé de deux parties, la transformée de Fourier en deux dimensions $e^{j2\pi(\eta x+\zeta y)}$ et la fonction de fenêtrage $\omega_M(x, y)$ (pour des blocs de $M \times M$) définie comme suit:

$$\omega_M(u, v) = \begin{cases} 1 & \text{si } |x| \leq M/2 \text{ et } |y| \leq M/2 \\ 0 & \text{autrement} \end{cases} \tag{4.1}$$

Ainsi la transformée fenêtrée de Fourier, $g_t(x, y)$, de l'image $I_t(x, y)$ est exprimée comme suit:

$$g_t(x, y) = h(x, y) \star I_t(x, y) = \int_{-\infty}^{\infty} \omega_M(x, y) I_t(x - r, y - s) e^{j2\pi(\eta r+\zeta s)}\, dr\, ds \tag{4.2}$$

$$= \int_{-M/2}^{M/2} \int_{-M/2}^{M/2} I_t(x - r, y - s) e^{j2\pi(\eta r+\zeta s)}\, dr\, ds \tag{4.3}$$

et la phase de la transformée, $\Phi_t(x, y)$, est calculée par:

$$\Phi_t(x, y) = arg[g_t(x, y)] \tag{4.4}$$

Weng (Weng 1993, Weng et al. 1992) propose d'apparier le point (x, y) de l'image au temps t avec le point $(x + v_1, y + v_2)$ de l'image au temps $t + 1$ si la différence de phase $E_\Phi(\vec{v}) = \Phi_{t+1}(x + v_1, y + v_2) - \Phi_t(x, y)$ est minimale.

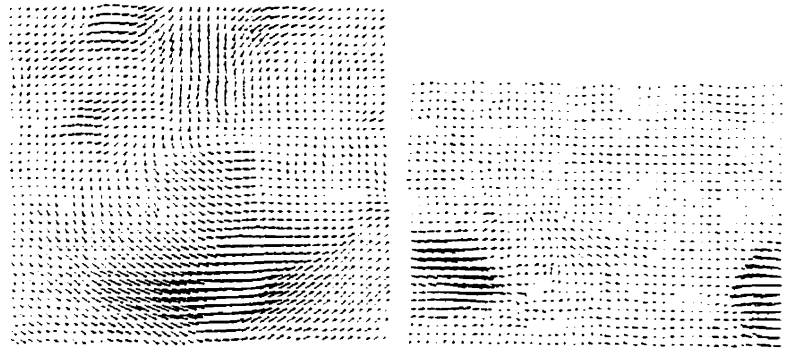
De plus, il impose une contrainte de douceur au champ de déplacement ($E_d(\vec{v}) = \|\vec{v}(x, y) - E[\vec{v}(x, y)]\|^2$), où $E[\vec{v}(x, y)]$ représente le déplacement moyen calculé à l'intérieur d'une région autour du pixel (x, y). Une architecture de calcul en multirésolution est utilisée pour effectuer le processus d'appariement. À chaque niveau de résolution, le vecteur de déplacement $\vec{v}(x, y)$ correspondant au point (x, y) est premièrement initialisé à la valeur de déplacement obtenue au niveau précédent. Ensuite, on determine un incrément $\vec{\Delta}$ de $\vec{v}$ de façon à minimiser une combinaison linéaire entre $E_\Phi(\vec{v} + \vec{\Delta})$ et $E_d(\vec{v} + \vec{\Delta})$:

$$[\Phi_{t+1}(x + v_1 + \Delta_1, y + v_2 + \Delta_2) - \Phi_t(x, y)]^2 + \lambda^2 \|\vec{v} + \vec{\Delta} - E[\vec{v}]\|^2 \qquad (4.5)$$

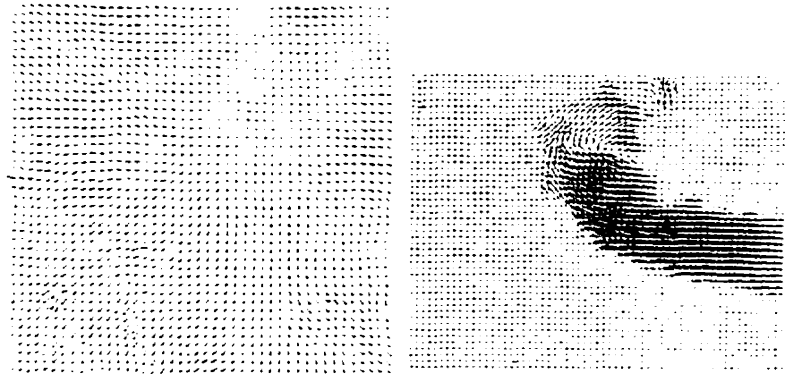
En utilisant l'expansion en série de Taylor autour de $(x + v_1, y + v_2)$, la valeur de $\vec{\Delta}$ qui minimise l'équation 4.5 est obtenue en faisant les dérivées partielles du résultat de l'expansion par rapport aux deux composantes de $\vec{\Delta}$.

Nous avons testé cette méthode avec les séquences de la figure 4.1. Les flux optiques calculés sont montrés à la figure 4.2.

Nous remarquons que des vecteurs de mouvements ont été calculés même aux régions statiques. Pour pallier à ce problème, nous avons comparé les deux images successives pour détecter ces régions statiques et ne calculer le flux optique qu'aux



(a) Séquence "Rubik" (b) Séquence "Taxi"



(c) Séquence "Nasa" (d) Séquence "Tunnel"

Figure 4.2 : Flux optique calculé avec la technique de Weng pour chacune des séquences types de la figure 4.1

régions dynamiques. Ceci a l'avantage aussi de diminuer le temps de calcul du flux optique. Les flux obtenus en faisant ce pré-traitement sont montrés à la figure 4.3. Le module de détection des blocs dynamiques sera expliqué en détail à la section 4.3.1.

(a) Séquence "Rubik" (b) Séquence "Taxi"

(c) Séquence "Nasa" (d) Séquence "Tunnel"

Figure 4.3 : Flux optique calculé aux régions dynamiques pour chacune des séquences types de la figure 4.1

4.3 Compression du mouvement

4.3.1 Détection des blocs statiques

Définition 4.1 *Les* **blocs statiques** *sont les blocs du plan image dont le contenu ne varie pas au cours du temps et qui peuvent par conséquent être approximés par leur contenu dans l'image précédente.*

D'après la définition même de ces blocs. il est évident qu'ils constituent une grande source de redondances dans les séquences d'images.

Pour détecter ces blocs, il suffit de comparer une mesure de distance entre les blocs de même position à deux instants successifs avec un seuil. et de décider que le bloc est statique lorsque la distance est inférieure au seuil. La mesure de distance la plus souvent utilisée est l'erreur quadratique (EQ). Notons par $B_t(x, y)$ le bloc de taille $\sqrt{N} \times \sqrt{N}$ au temps t qui commence à la position (x, y), l'erreur quadratique correspondant à ce bloc peut être exprimée par la relation:

$$EQ(B) = \sqrt{\sum_{i=x}^{x+\sqrt{N}} \sum_{j=y}^{y+\sqrt{N}} (I_{t+1}(i,j) - I_t(i,j))^2} \tag{4.6}$$

Il est important de choisir la valeur du seuil indépendamment des images de test. Il faut donc comparer l'erreur quadratique d'un bloc avec une mesure statistique calculée sur l'erreur quadratique de tous les autres blocs de l'image pour pouvoir décider si le bloc est statique ou dynamique.

Puisque les erreurs quadratiques calculées pour les blocs sont bruitées[1], la moyenne de ces erreurs semble être un bon choix pour le seuil car la moyenne est un filtre passe bas peu sensible au bruit. Ce choix signifie que tous les blocs ayant une erreur supérieure à la moyenne globale de cette erreur sur l'image sont des blocs dynamiques. Le seuil peut donc être calculé selon l'équation:

[1] À cause du bruit dans l'acquisition des images

$$\text{Seuil} = E[EQ(B)] \tag{4.7}$$

En se basant sur ce critère pour éliminer les blocs statiques des flux montrés à la figure 4.2, nous obtenons les flux à la figure 4.3. Nous remarquons dans cette figure que la mesure précédente peut nous éliminer des blocs dynamiques dans les scènes où il y a beaucoup de mouvement. Un exemple d'une telle situation est la séquence "Nasa" où toute l'image bouge. Pour pallier à ce problème, nous avons écrêté le seuil précédent par une valeur maximale de l'EQ désirée ($EQ_{\max}$). Ceci peut être exprimé par l'équation suivante:

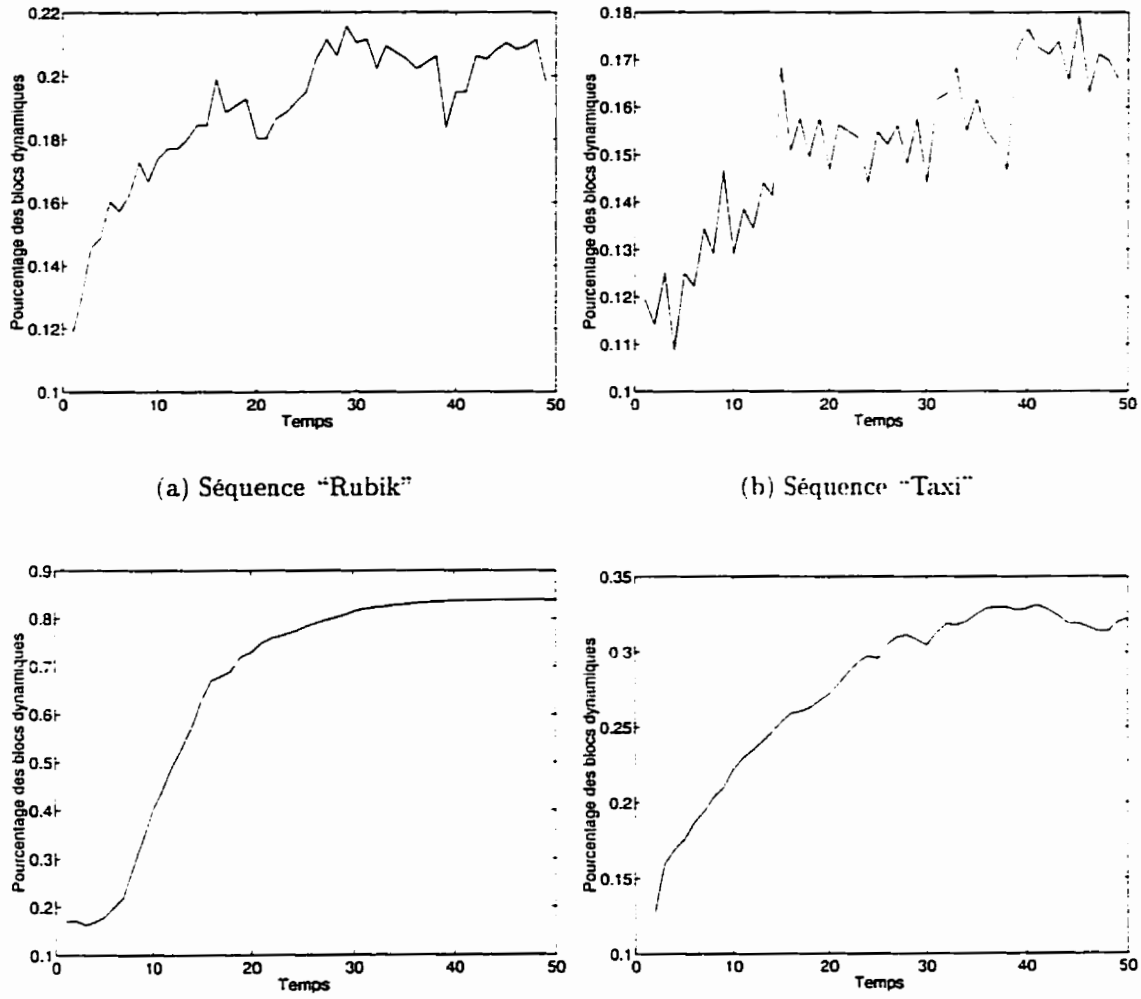
$$\text{Seuil} = \min\left(E[EQ(B)], EQ_{\max}\right) \tag{4.8}$$

Nous avons choisit $EQ_{\max} = 8 \times \sqrt{N}$ [2] qui correspond à une erreur quadratique moyenne de 8 pour des blocs de $\sqrt{N} \times \sqrt{N}$. Ceci revient à considérer tous les blocs ayant une EQM supérieure à 8 comme étant dynamiques. Ce choix doit nous permettre d'identifier les objets en mouvement comme étant dynamiques tout en négligeant le bruit dans l'acquisition des images comme source de mouvement. La valeur choisie semble réaliser ce compromis et donner des résultats à notre satisfaction comme expliqué plus loin.

A la figure 4.4, nous avons la variation du pourcentage de blocs dynamiques dans le

[2] $EQM^2 = \frac{EQ^2}{N}$

temps. Nous remarquons que l'adaptation du seuil nous permet d'augmenter linéairement dans le temps le pourcentage de blocs dynamiques de la séquence "Nasa". D'autre part, le pourcentage de blocs dynamiques des séquences "Rubik", "Taxi" et "Tunnel" se stabilise à 21%, 16% et 32% respectivement.



(a) Séquence "Rubik"

(b) Séquence "Taxi"

(c) Séquence "Nasa"

(d) Séquence "Tunnel"

Figure 4.4 : Variation du pourcentage de blocs dynamiques dans le temps

Au bout de 10 images, nous obtenons les flux optiques de la figure 4.5. Nous remarquons que les objets en mouvement ont été détectés comme étant dynamiques. D'autre part, une grande partie du flux correspondant aux régions statiques de l'image a été éliminée ce qui nous permet de diminuer le temps de calcul du flux et de diminuer le volume de l'information de mouvement à transmettre. Il est à noter que des régions en mouvement peuvent être considérées statiques si leurs intensités lumineuses sont uniformes. Ceci est normal puisque leur mouvement n'apporte pas un changement aux intensités lumineuses dans les images. Un exemple d'une telle situation peut être perçu dans la séquence "Tunnel" où plusieurs endroits du train sont considérés comme étant statiques.

Au tableau 4.2, nous avons l'erreur introduite par le fait de remplacer, dans la deuxième image de la séquence, tous les blocs statiques par leur contenu dans l'image précédente. Cette erreur est représentée par l'erreur quadratique moyenne (EQM) et le rapport signal à bruit crête à crête (RSBC). Ce dernier est défini par l'équation:

$$RSBC = 10 \times log_{10}(\frac{255^2}{EQM^2}) \tag{4.9}$$

Nous remarquons que cette erreur est plus grande pour les séquences "Taxi" et "Nasa" que pour les deux autres séquences. Ceci est principalement causé par des blocs en mouvement qui ont été considérés comme étant statiques. Ceci est évident dans le cas de la séquence "Nasa" où le mouvement est global. Pour la séquence "Taxi", une grande partie de la voiture au bas gauche de l'image a été considérée

(a) Séquence "Rubik" (b) Séquence "Taxi"

(c) Séquence "Nasa" (d) Séquence "Tunnel"

Figure 4.5 : Flux optique calculé aux régions dynamiques au bout de 10 images pour chacune des séquences types de la figure 4.1

statique causant une erreur non négligeable. Ces erreurs sont diminuées dans le temps en adaptant le seuil de détection des blocs dynamiques comme expliqué auparavant.

4.3.2 Compression du mouvement des blocs dynamiques

Suite à l'élimination des blocs statiques, le mouvement dense des blocs dynamiques doit être compressé. Nous nous sommes basé sur la représentation du chapitre 3 pour

Tableau 4.2 : Erreur introduite par le fait de remplacer, dans la deuxième image de la séquence, tous les blocs statiques par leur contenu dans l'image précédente

Séquence	EQM	RSBC
"Rubik"	1.56	44 dB
"Taxi"	3.09	38 dB
"Nasa"	3.37	37.5 dB
"Tunnel"	2.27	41 dB

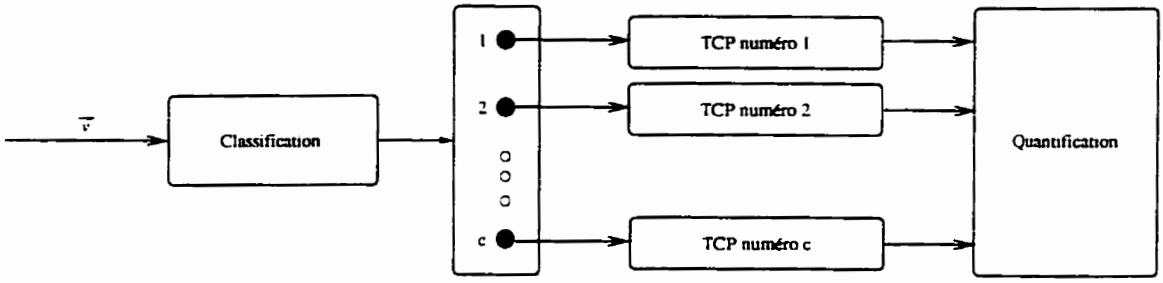


Figure 4.6 : Structure générale de l'algorithme de compression du mouvement

faire cette compression. La structure générale de l'algorithme de compression est celle de la figure 4.6. Le mouvement de chaque bloc dynamique est classé dans une des c sources. Par la suite, le coefficient principal de l'amplitude et de la phase du mouvement de ce bloc sont calculés en effectuant le produit scalaire de ces derniers avec la composante principale correspondante de la source choisie. Enfin, les coefficients quantifiés avec l'information de quantification qui constituent la représentation du mouvement du bloc sont transmis au récepteur.

Dans ce qui suit, nous allons décrire les deux modules principaux de cette compression: la classification et la transformation en composantes principales.

4.3.2.1 Classification du mouvement

Définition 4.2 *Un bloc est attribué à une classe s'il ressemble plus aux blocs typiques générés par la source de cette classe qu'aux blocs typiques générés par les autres sources.*

Cette définition exige implicitement l'extraction d'un bloc typique b_i par source i et le choix d'une mesure de ressemblance entre ces blocs typiques et le bloc b courant $r(b, b_i)$. Ceci peut être écrit sous forme mathématique comme:

$$b \in i \text{ si } r(b, b_i) > r(b, b_j) \quad \forall j \neq i \tag{4.10}$$

Il est évident, par la définition même de la composante principale d'une source, qu'elle est celle qui représente le mieux les blocs générés par cette dernière. Nous avons, par conséquent, choisi le produit scalaire entre le bloc et la composante principale d'une source comme mesure de ressemblance.

Comme nous l'avons expliqué au chapitre 3, l'amplitude a et la phase θ d'un bloc doivent être classés indépendamment. L'algorithme de classification peut donc être résumé par les équations suivantes:

$$a \in i \text{ si } T_i^a(1).\,\widetilde{a} > T_j^a(1).\,\widetilde{a} \quad \forall j \neq i \tag{4.11}$$

$$\theta \in i \text{ si } T_i^\theta(1).\,\widetilde{\theta} > T_j^\theta(1).\,\widetilde{\theta} \quad \forall j \neq i \tag{4.12}$$

où $\widetilde{a}$ et $\widetilde{\theta}$ sont les blocs d'amplitude et de phase représentés en forme de vecteurs comme décrit à la section 3.2 et $T_i^a(1)$ et $T_i^\theta(1)$ correspondent à la composante principale de l'amplitude et de la phase respectivement de la $i^{\text{ème}}$ source.

Dans ce qui suit, nous avons les résultats de classification de chacune des séquences de test de la figure 4.1. L'amplitude et la phase du flux optique calculé sont illustrés à la figure 4.7 en niveaux de gris, les valeurs plus élevées étant plus claires.

Aux figures 4.8 à 4.11, nous avons les composantes principales de l'amplitude et de la phase apprises pour chacune des séquences. Le nombre de sources a été fixé à huit. Dans la section 4.3.2.2, nous discuterons du choix à faire pour le nombre de sources. Ces composantes sont représentées pour la source i par les matrices $\overset{*}{T}_i^a$ et $\overset{*}{T}_i^\theta$ de taille $\sqrt{N} \times \sqrt{N}$ (pour des blocs de même taille). Ces matrices correspondent à la transformation inverse à celle effectuée sur les blocs d'amplitude et de phase du flux pour les représenter en terme de vecteurs et peuvent par conséquent être décrites par:

$$\overset{*}{T}_i^a(x,y) = T_i^a(1, \sqrt{N}[y-1]+x) \tag{4.13}$$

$$\overset{*}{T}_i^\theta(x,y) = T_i^\theta(1, \sqrt{N}[y-1]+x) \tag{4.14}$$

où $T_i^a(1,u)$ et $T_i^\theta(1,u)$ correspondent au $u^{\text{ème}}$ élément des vecteurs $T_i^a(1)$ et $T_i^\theta(1)$.

La classification peut être interprétée comme un problème de reconnaissance de forme. Prenons l'exemple de la séquence "Rubik". Chacun des blocs de la figure 4.7(a)

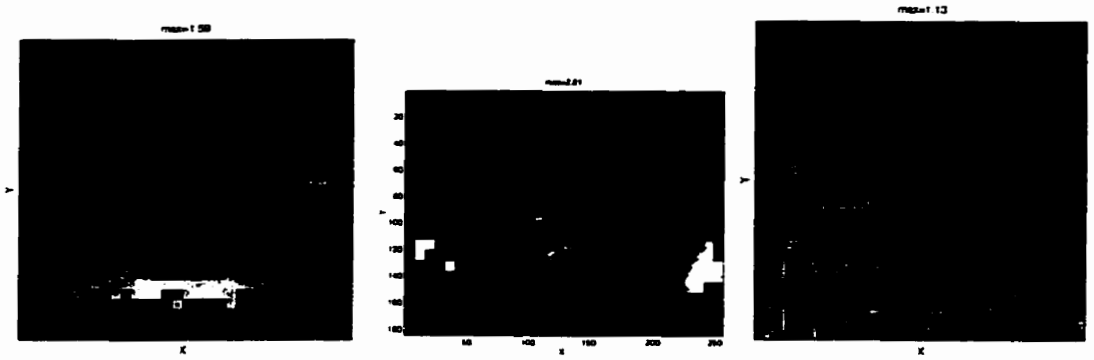
est associé à la source dans les figures 4.8(a) à 4.8(h) qui a la matrice $\overset{*}{T}_i^a\ (x, y)$ lui ressemblant le plus.

Le résultat de la classification est donné aux figures 4.12 à 4.15. Afin de mieux comprendre ces résultats, nous allons les interpréter, sans perte de généralité, sur un exemple de la séquence "Rubik". A la figure 4.16, nous avons tous les blocs d'amplitude de la séquence "Rubik" qui ont été générés par la première source et la mesure de ressemblance pour chacun de ces blocs avec les huit sources. Nous remarquons que tous les blocs d'amplitude générés par la première source ont tendance à avoir une amplitude plus grande dans la partie supérieure du bloc. Ceci est une propriété visible sur la composante principale de la source (figure 4.8(a)). De plus, nous remarquons que les composantes principales de la $4^{\text{ème}}$ et la $5^{\text{ème}}$ source (figure 4.8(d) et 4.8(e) respectivement) partagent aussi cette propriété ce qui explique la mesure de ressemblance élevée pour ces deux autres sources et qui entraine une compétition entre la première source et ces deux sources.

Nous remarquons dans les figures 4.12 à 4.15 que plusieurs sources se trouve à ne générer aucun bloc. Nous les dénomons "sources mortes" puisqu'aucun bloc n'est attribué à ces sources et qu'elles n'ont pas l'opportunité de s'adapter au cours du temps. Dans l'implantation actuelle, ces sources ont été détectées et leur composante principale a été remplaçée par celle de la source générant le plus de blocs, qu'on appelle la source dominante. A l'image suivante, tous les blocs qui auraient dû être générés par la source dominante auront une probabilité égale d'être générés par cette

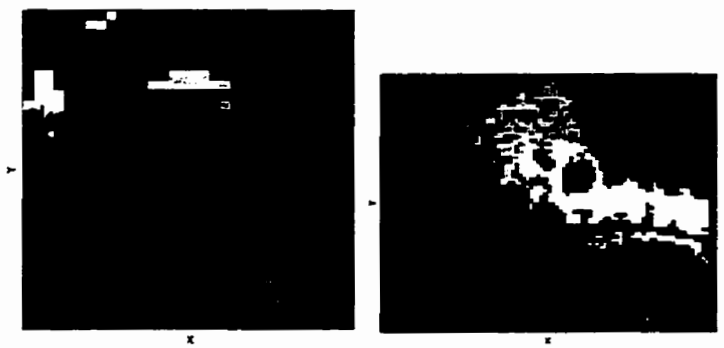
dernière ou par la source morte (puisqu'elles ont la même composante principale) d'où l'élimination de la source morte.

Il est à noter que la classification obtenue ne correspond pas à la classification que le système visuel humain aurait fait pour le mouvement de la scène. Cette classification est, tout de même, optimale du point de vue codage puisqu'elle associe chaque bloc à la classe qui conserve le plus de l'énergie de ce bloc dans sa composante principale.



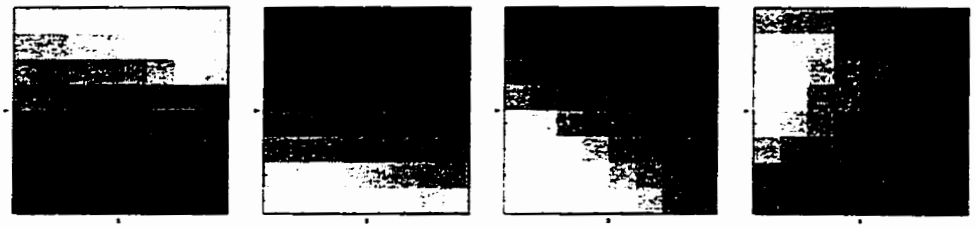
(a) Amplitude de "Rubik" (b) Amplitude de "Taxi" (c) Amplitude de "Nasa"

(d) Amplitude de "Tunnel" (e) Phase de "Rubik" (f) Phase de "Taxi"

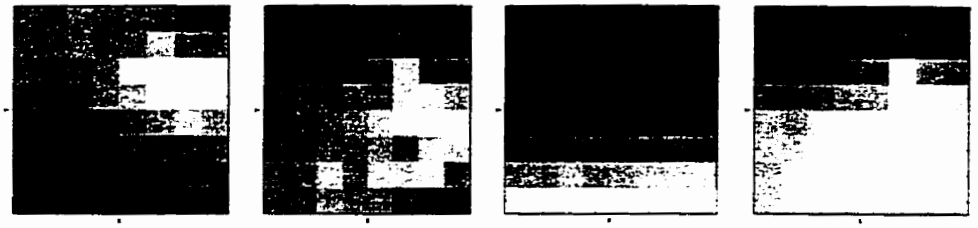


(g) Phase de "Nasa" (h) Phase de "Tunnel"

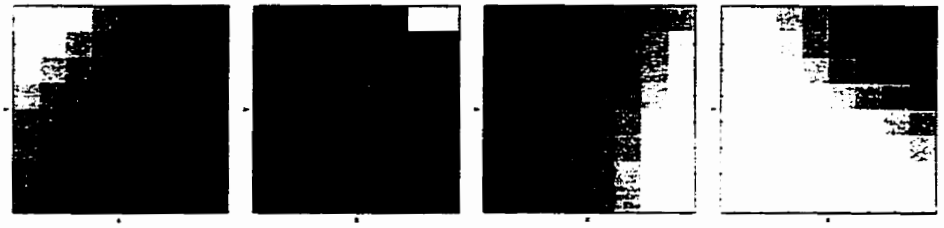
Figure 4.7 : Amplitude et phase du flux optique calculé aux régions dynamiques pour chacune des séquences types de la figure 4.1



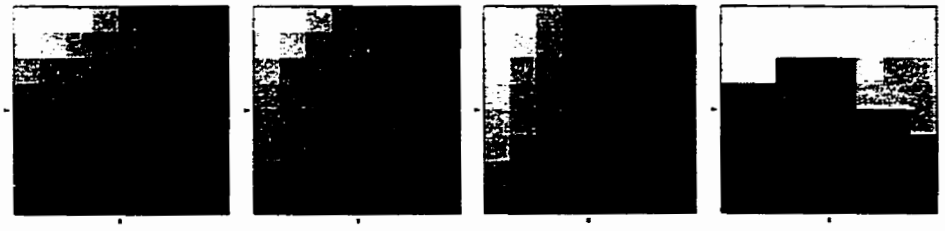
(a) Source 1 d'amplitude (b) Source 2 d'amplitude (c) Source 3 d'amplitude (d) Source 4 d'amplitude



(e) Source 5 d'amplitude (f) Source 6 d'amplitude (g) Source 7 d'amplitude (h) Source 8 d'amplitude

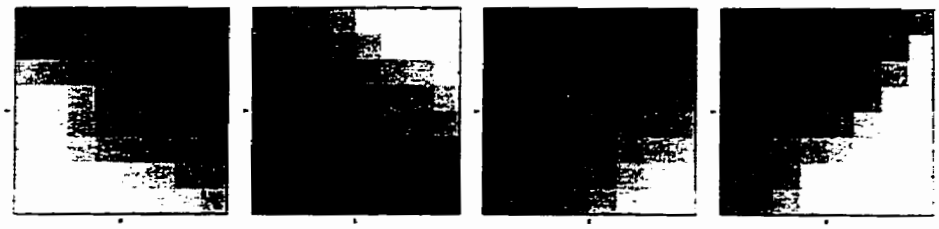


(i) Source 1 de la phase (j) Source 2 de la phase (k) Source 3 de la phase (l) Source 4 de la phase

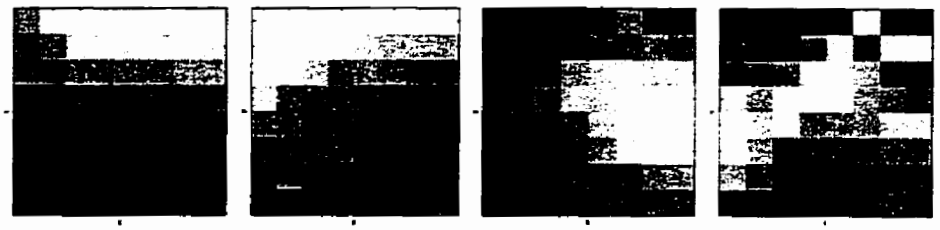


(m) Source 5 de la phase (n) Source 6 de la phase (o) Source 7 de la phase (p) Source 8 de la phase

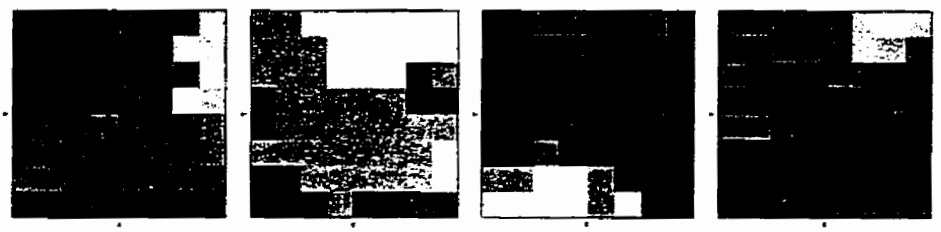
Figure 4.8 : Composantes principales de l'amplitude et de la phase des sources de la séquence "Rubik"



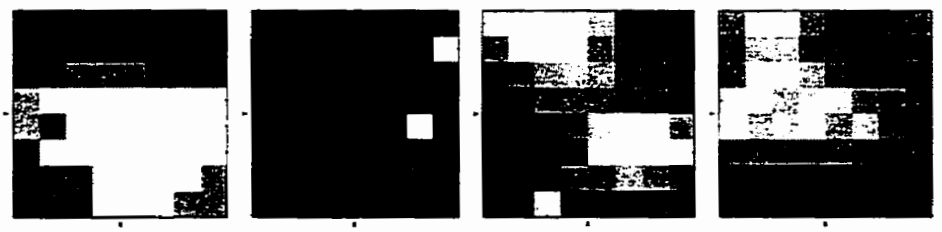
(a) Source 1 de l'amplitude (b) Source 2 de l'amplitude (c) Source 3 de l'amplitude (d) Source 4 de l'amplitude



(e) Source 5 de l'amplitude (f) Source 6 de l'amplitude (g) Source 7 de l'amplitude (h) Source 8 de l'amplitude

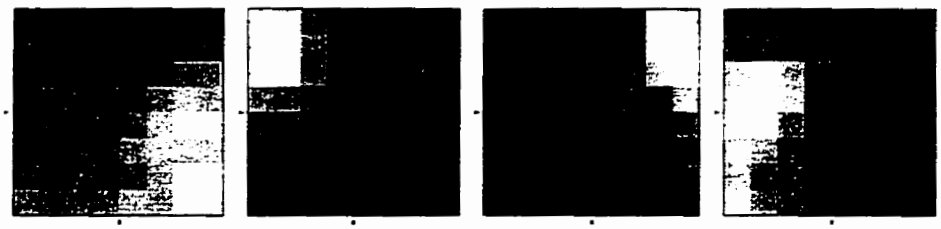


(i) Source 1 de la phase (j) Source 2 de la phase (k) Source 3 de la phase (l) Source 4 de la phase

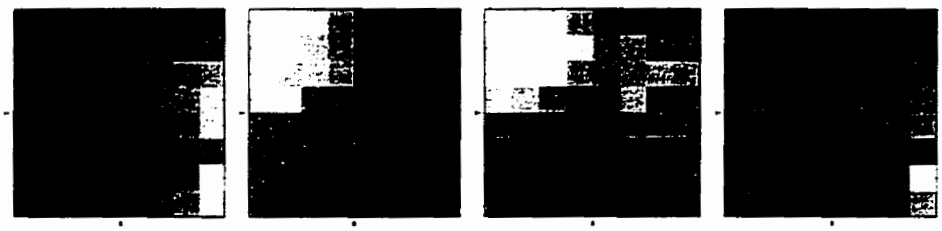


(m) Source 5 de la phase (n) Source 6 de la phase (o) Source 7 de la phase (p) Source 8 de la phase

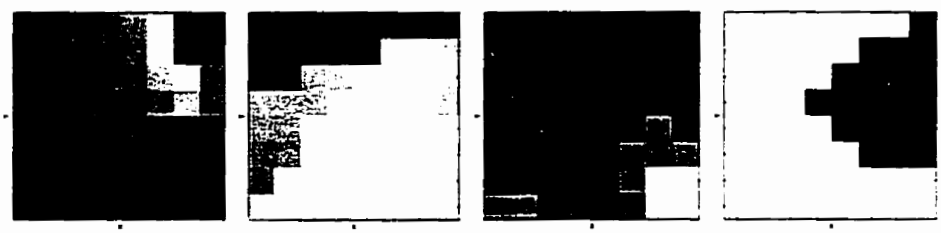
Figure 4.9 : Composantes principales de l'amplitude et de la phase des sources de la séquence "Taxi"



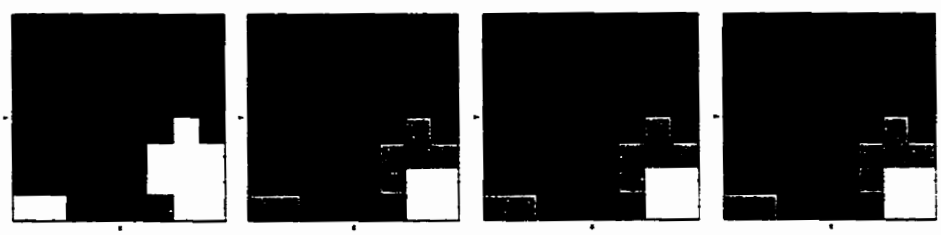
(a) Source 1 de l'amplitude (b) Source 2 de l'amplitude (c) Source 3 de l'amplitude (d) Source 4 de l'amplitude



(e) Source 5 de l'amplitude (f) Source 6 de l'amplitude (g) Source 7 de l'amplitude (h) Source 8 de l'amplitude

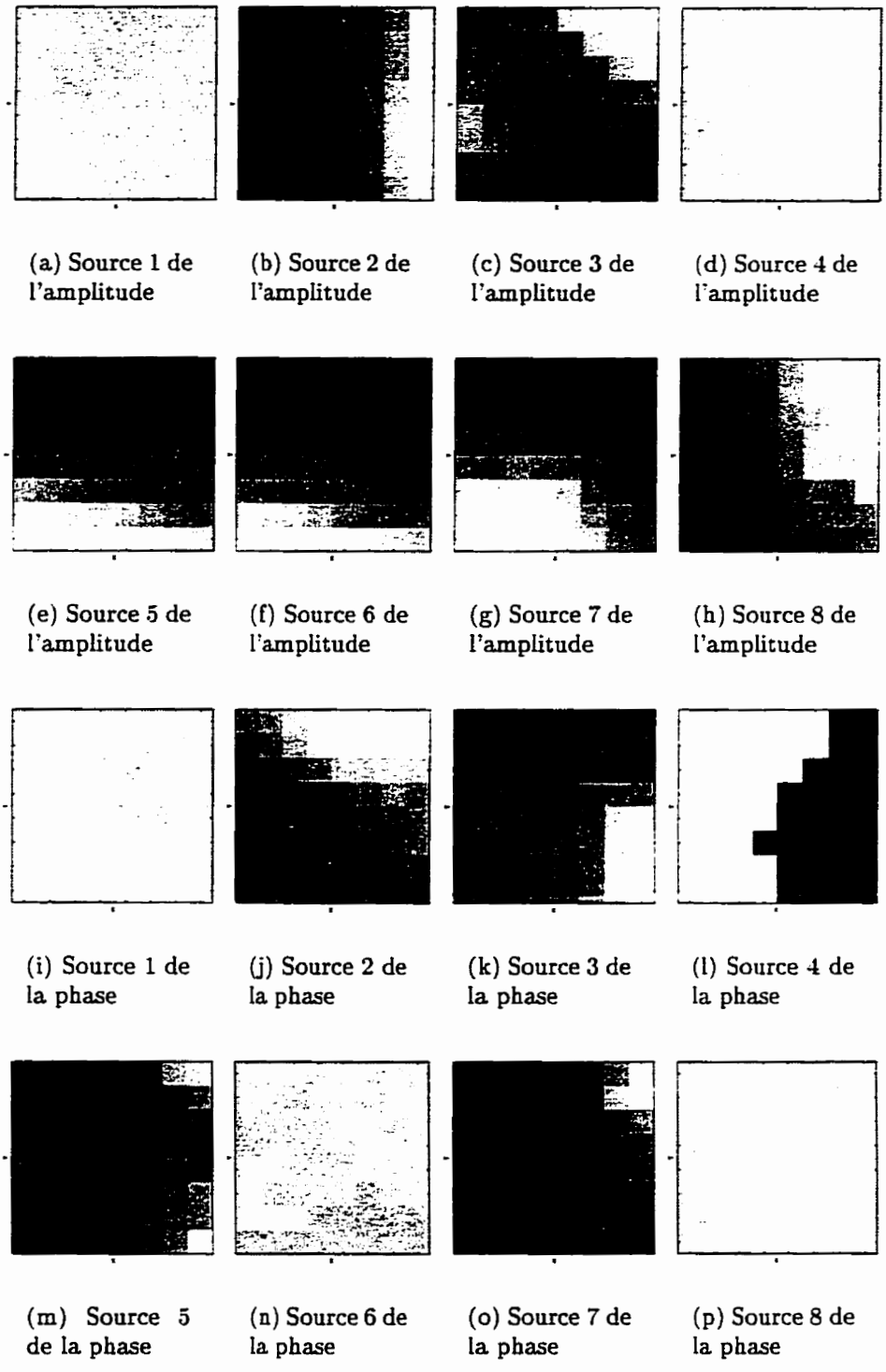


(i) Source 1 de la phase (j) Source 2 de la phase (k) Source 3 de la phase (l) Source 4 de la phase



(m) Source 5 de la phase (n) Source 6 de la phase (o) Source 7 de la phase (p) Source 8 de la phase

Figure 4.10 : Composantes principales de l'amplitude et de la phase des sources de la séquence "Nasa"



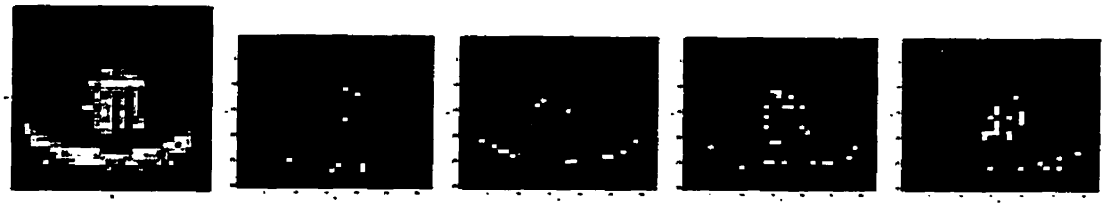
(a) Source 1 de l'amplitude (b) Source 2 de l'amplitude (c) Source 3 de l'amplitude (d) Source 4 de l'amplitude

(e) Source 5 de l'amplitude (f) Source 6 de l'amplitude (g) Source 7 de l'amplitude (h) Source 8 de l'amplitude

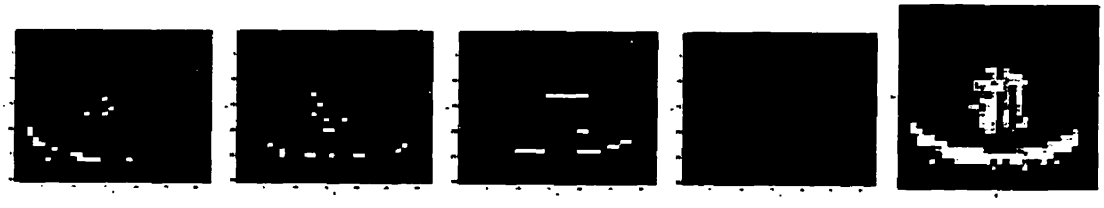
(i) Source 1 de la phase (j) Source 2 de la phase (k) Source 3 de la phase (l) Source 4 de la phase

(m) Source 5 de la phase (n) Source 6 de la phase (o) Source 7 de la phase (p) Source 8 de la phase

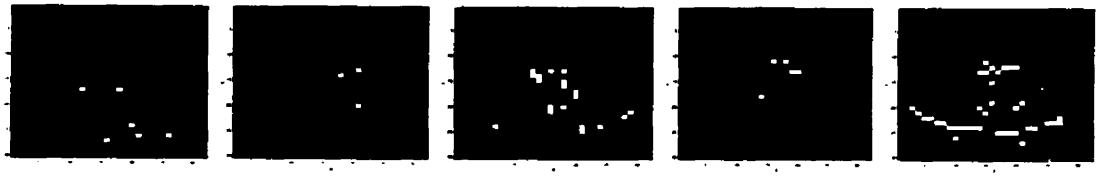
Figure 4.11 : Composantes principales de l'amplitude et de la phase des sources de la séquence "Tunnel"



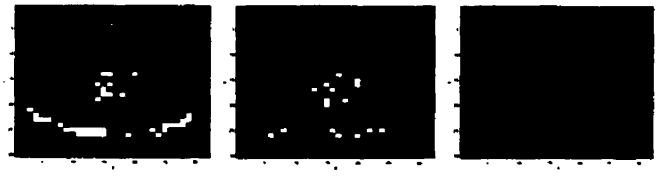
(a) Classification de l'amplitude (b) Source 1 de l'amplitude (c) Source 2 de l'amplitude (d) Source 3 de l'amplitude (e) Source 4 de l'amplitude



(f) Source 5 de l'amplitude (g) Source 6 de l'amplitude (h) Source 7 de l'amplitude (i) Source 8 de l'amplitude (j) Classification de la phase

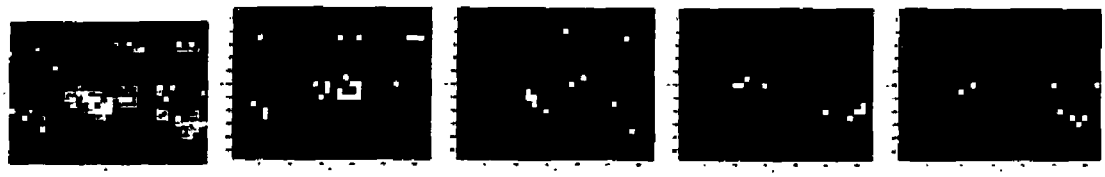


(k) Source 1 de la phase (l) Source 2 de la phase (m) Source 3 de la phase (n) Source 4 de la phase (o) Source 5 de la phase



(p) Source 6 de la phase (q) Source 7 de la phase (r) Source 8 de la phase

Figure 4.12 : Classification de l'amplitude et de la phase de la séquence "Rubik"



(a) Classification de l'amplitude

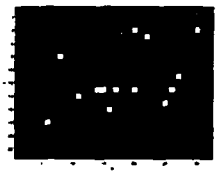
(b) Source 1 de l'amplitude

(c) Source 2 de l'amplitude

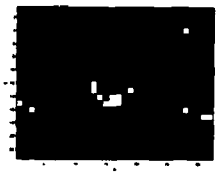
(d) Source 3 de l'amplitude

(e) Source 4 de l'amplitude

(f) Source 5 de l'amplitude



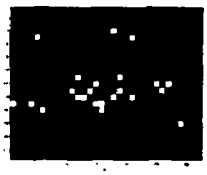
(g) Source 6 de l'amplitude



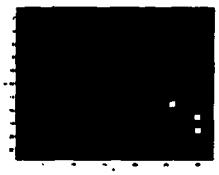
(h) Source 7 de l'amplitude

(i) Source 8 de l'amplitude

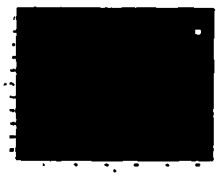
(j) Classification de la phase



(k) Source 1 de la phase

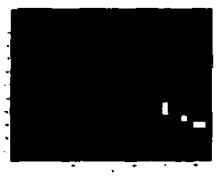


(l) Source 2 de la phase



(m) Source 3 de la phase

(n) Source 4 de la phase



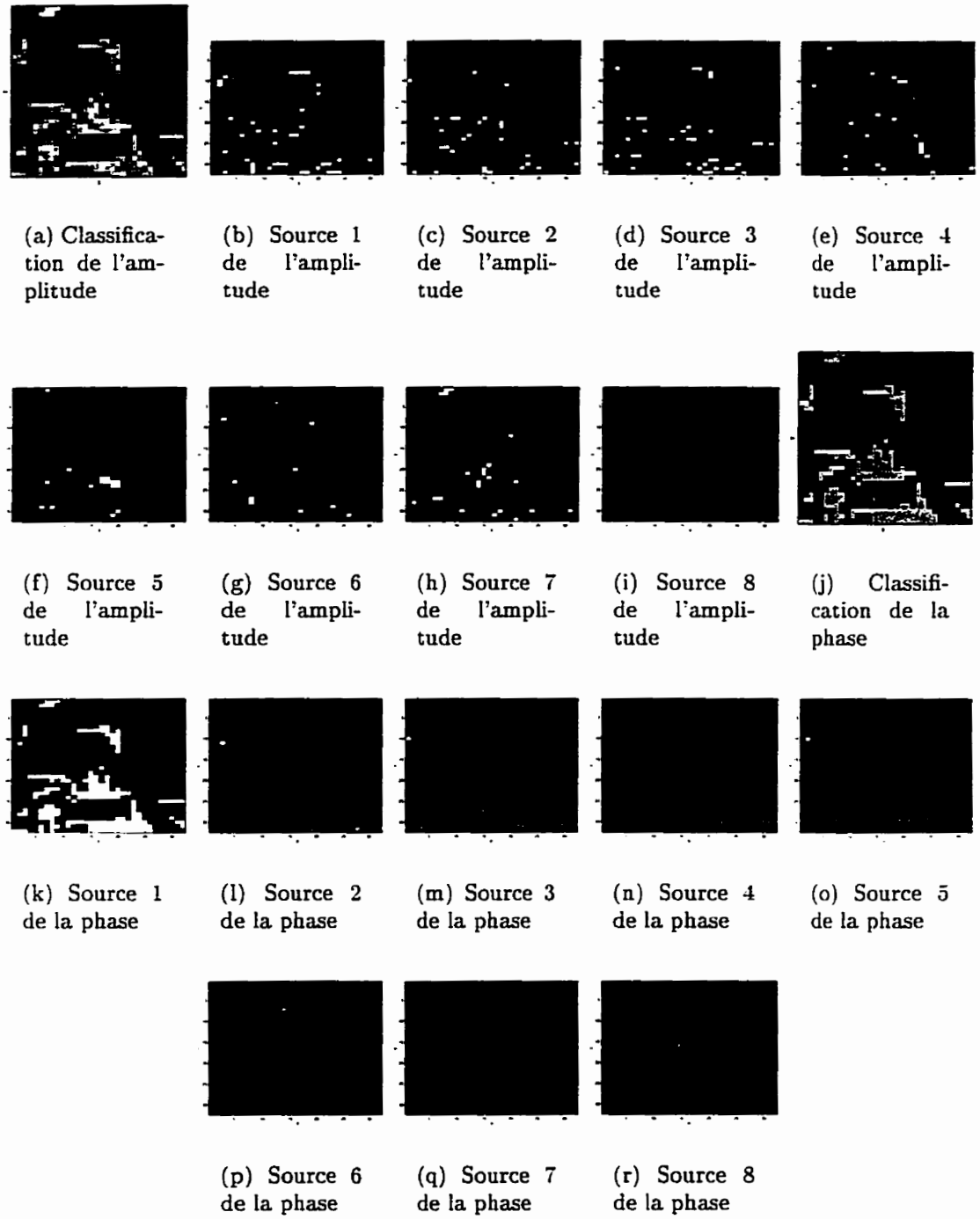
(o) Source 5 de la phase

(p) Source 6 de la phase

(q) Source 7 de la phase

(r) Source 8 de la phase

Figure 4.13 : Classification de l'amplitude et de la phase de la séquence "Taxi"



(a) Classification de l'amplitude (b) Source 1 de l'amplitude (c) Source 2 de l'amplitude (d) Source 3 de l'amplitude (e) Source 4 de l'amplitude

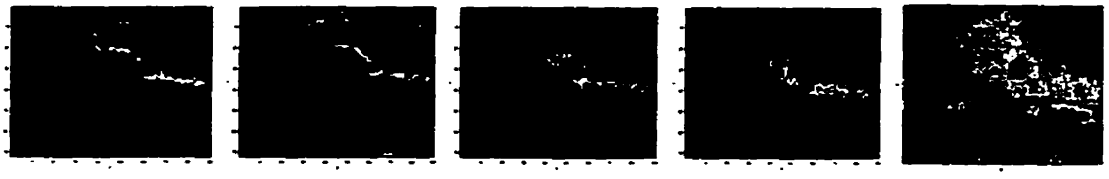
(f) Source 5 de l'amplitude (g) Source 6 de l'amplitude (h) Source 7 de l'amplitude (i) Source 8 de l'amplitude (j) Classification de la phase

(k) Source 1 de la phase (l) Source 2 de la phase (m) Source 3 de la phase (n) Source 4 de la phase (o) Source 5 de la phase

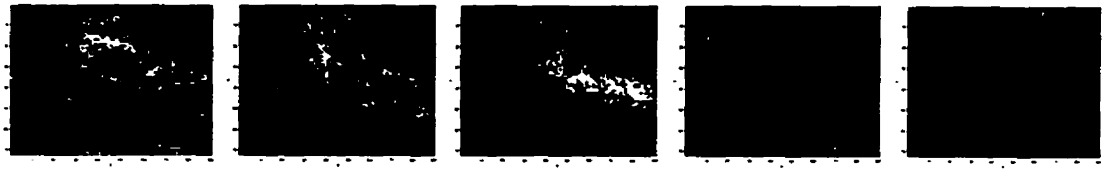
(p) Source 6 de la phase (q) Source 7 de la phase (r) Source 8 de la phase

Figure 4.14 : Classification de l'amplitude et de la phase de la séquence "Nasa"

(a) Classification de l'amplitude (b) Source 1 de l'amplitude (c) Source 2 de l'amplitude (d) Source 3 de l'amplitude (e) Source 4 de l'amplitude



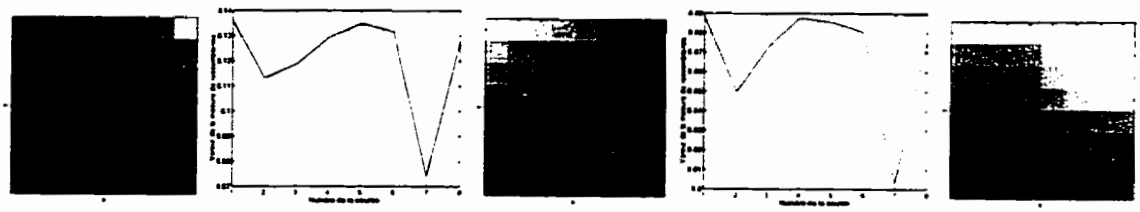
(f) Source 5 de l'amplitude (g) Source 6 de l'amplitude (h) Source 7 de l'amplitude (i) Source 8 de l'amplitude (j) Classification de la phase



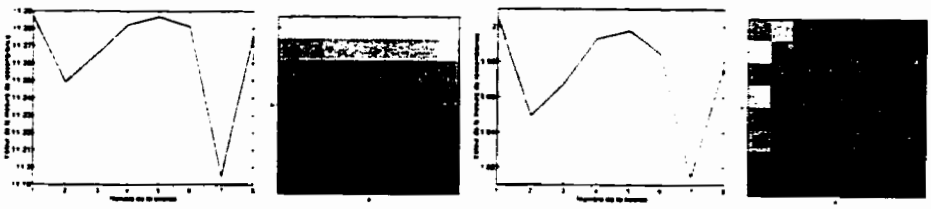
(k) Source 1 de la phase (l) Source 2 de la phase (m) Source 3 de la phase (n) Source 4 de la phase (o) Source 5 de la phase

(p) Source 6 de la phase (q) Source 7 de la phase (r) Source 8 de la phase

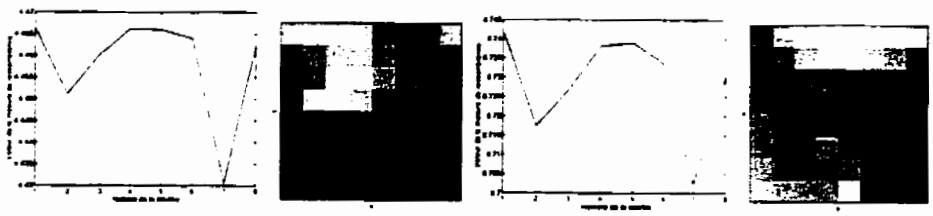
Figure 4.15 : Classification de l'amplitude et de la phase de la séquence "Tunnel"



(a) Bloc 1 (b) Ressemblance du bloc 1 (c) Bloc 2 (d) Ressemblance du bloc 2 (e) Bloc 3



(f) Ressemblance du bloc 3 (g) Bloc 4 (h) Ressemblance du bloc 4 (i) Bloc 5



(j) Ressemblance du bloc 5 (k) Bloc 6 (l) Ressemblance du bloc 6 (m) Bloc 7

(n) Ressemblance du bloc 7 (o) Bloc 8 (p) Ressemblance du bloc 8

Figure 4.16 : Les huit blocs d'amplitude de la séquence "Rubik" qui sont générés par la première classe et leur mesure de ressemblance associées

4.3.2.2 Compression du mouvement dans chacune des classes

Une fois le mouvement classé, les coefficients principaux de l'amplitude et de la phase de chaque bloc dynamique sont calculés en effectuant le produit scalaire de ces derniers par les composantes principales respectives des sources choisies c_a et c_θ selon les équations:

$$y^a_{c_a}(1) = T^a_{c_a}(1)\ \tilde{a} \tag{4.15}$$

$$y^\theta_{c_\theta}(1) = T^\theta_{c_\theta}(1)\ \tilde{\theta} \tag{4.16}$$

$$\tag{4.17}$$

Par la suite, les composantes principales sont adaptées selon l'apprentissage Hebbien et seront utilisées pour classer et coder le mouvement de l'image suivante.

Les coefficients principaux de l'amplitude et de la phase des blocs dynamiques sont quantifiés et transmis au récepteur avec la classification de ces blocs. Ce dernier restitue ces coefficients ($\hat{y}^a_{c_a}(1)$ et $\hat{y}^\theta_{c_\theta}(1)$) par quantification inverse et les multiplie par la composante principale respective de leur source (adaptée à l'image précédente) pour reconstruire le mouvement dense du bloc selon les équations:

$$\hat{a} = \hat{y}^a_{c_a}(1)\ \overset{*}{T}{}^a_{c_a} \tag{4.18}$$

$$\hat{\theta} = \hat{y}^\theta_{c_\theta}(1)\ \overset{*}{T}{}^\theta_{c_\theta} \tag{4.19}$$

Cette reconstruction est aussi effectuée à l'émetteur pour permettre à ce dernier d'avoir la même approximation du mouvement que le récepteur. Ceci est important pour pouvoir utiliser le même mouvement à l'émetteur qu'au récepteur dans l'adaptation des composantes principales et la reconstruction des images.

Les composantes principales de chacune des sources sont adaptées à l'émetteur et au récepteur en se basant sur le mouvement reconstruit. L'algorithme de Sanger (1989) a été utilisé pour cette adaptation. A la section 3.3, nous avons démontré que les matrices de transformation apprises avec cet algorithme apporte une amélioration de qualité très significative[3] sur le mouvement reconstruit par rapport aux matrices fixes de la TCD. Le modèle d'apprentissage de Sanger peut être exprimé selon l'équation:

$$\Delta T = \eta(y^t x - TI[y^t y]T) \tag{4.20}$$

Ce modèle d'apprentissage permet de mettre à jour toutes les composantes des sources. Dans l'implantation actuelle, nous n'utilisons que la composante principale pour le codage. Par conséquent, nous aurions pu utiliser l'algorithme d'Oja (1982) expliqué à la section 3.3 (équation 3.17) pour adapter la composante principale des sources. Ceci aurait l'avantage de diminuer le temps de calcul pour l'adaptation car cet algorithme calcule uniquement la composante principale. Par souci de généralité, nous avons tout de même utilisé l'algorithme de Sanger pour adapter toutes les

[3] 558% et 372% sur la reconstruction d'un mouvement synthétique divergent et rotationnel respectivement

composantes . En effet, notre logiciel de simulation nous permet de faire des tests en gardant plus de coefficients que le coefficient principal, chose qui ne serait pas possible avec le modèle d'Oja.

Nous plaçons tous les blocs appartenant à une même classe dans un même lot. Des blocs sont choisis au hazard du lot pour adapter les composantes principales de la source. Le nombre de blocs choisi a été fixé à 10 fois le nombre de blocs dans le lot. En d'autres mots, chaque bloc est utilisé 10 fois dans l'adaptation de la source qui le génère. Il aurait été préférable d'adapter ce nombre pour arrêter l'apprentissage quand les modifications effectuées aux composantes principales deviennent inférieures à un seuil. Afin que tous les blocs d'un lot soit utilisés 10 fois dans l'adaptation, les blocs choisis sont retirés du lot au fur et à mesure. Quand le lot devient vide, tous les blocs sont remis dedans à nouveaux. Pour s'assurer de la convergence du système, nous avons initialisé le taux d'apprentissage à une grande valeur (0.2). Cette valeur est diminuée dans le temps selon la relation suivante[4]:

$$\eta(t) = \frac{1}{10t}, \quad t \neq 0 \tag{4.21}$$

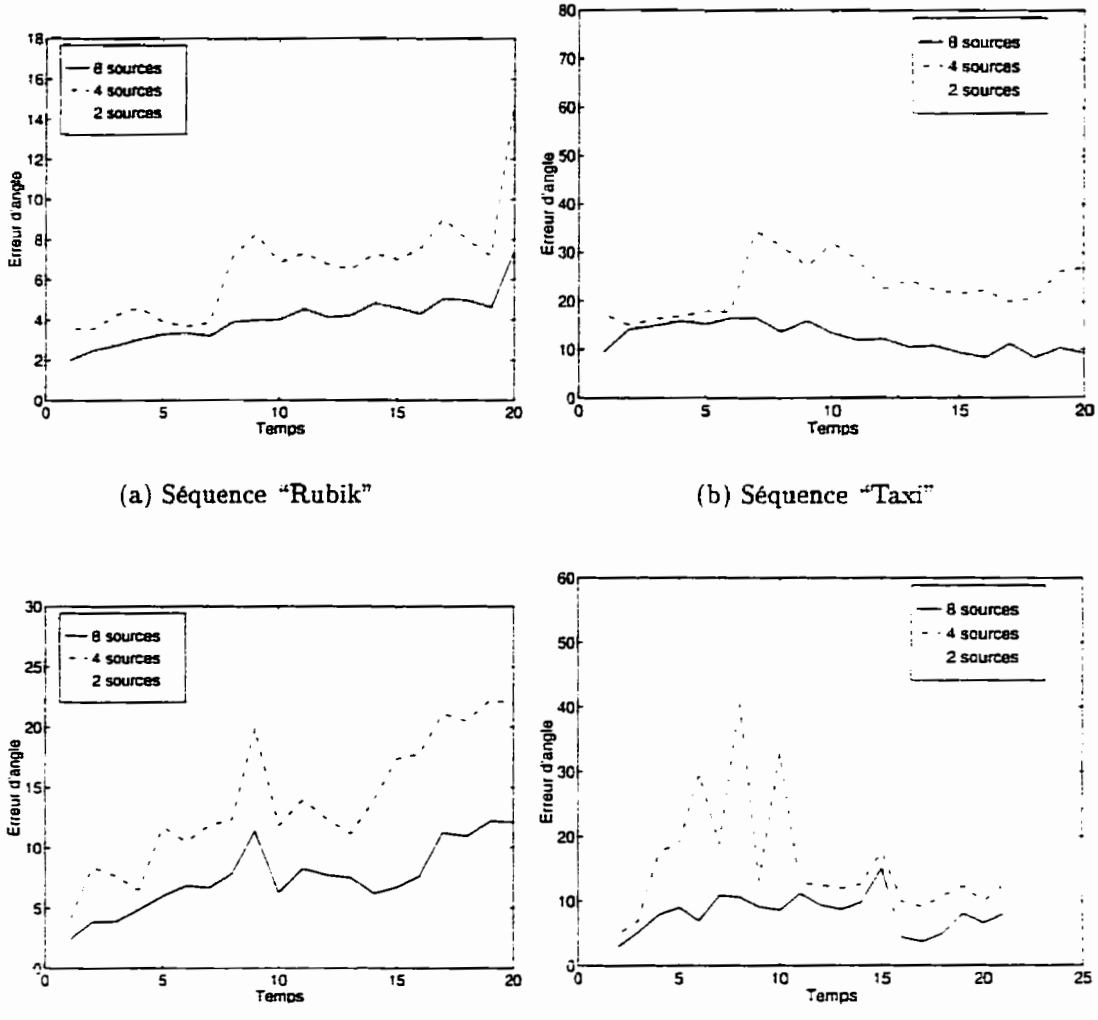
Par exemple, pour un lot de 100 blocs, nous aurons 1000 blocs à choisir et le taux d'apprentissage sera de 0.2 pour le premier bloc choisi et de $\frac{1}{10000}$ pour le dernier.

A la section 4.3.2.1, nous avons présenté les résultats de classification avec 8

[4]D'après la définition des séries harmoniques, cette relation vérifie les deux conditions de convergence ($\lim_{t\to\infty} \eta(t) = 0$ et $\sum_{t=0}^{\infty} \eta(t) = \infty$)

sources. Quand le nombre de sources augmente, l'information de classification devient plus volumineuse à transmettre. Par contre, plus le nombre de sources est élevé, plus la probabilité de trouver une source qui décrit bien chaque bloc augmente. Par conséquent, l'erreur d'angle devrait diminuer en augmentant le nombre de sources. La figure 4.17 confirme cette remarque. Dans cette figure, nous avons l'erreur d'angle dans le temps pour chacune des séquences de la figure 4.1 en utilisant 8,4 et 2 sources. La possibilité d'utiliser plus de sources que 8 n'a pas été investiguée car la mémoire disponible sur les ordinateurs utilisés ne l'a pas permis. Il est évident que la classe d'un bloc peut être représentée avec 3, 2 ou 1 bits quand nous utilisons 8, 4 ou 2 sources respectivement pour faire la classification. Par conséquent, le coût associé à l'augmentation de qualité en augmentant le nombre de sources est de 1 bit par bloc dynamique à chaque fois que nous augmentons le nombre de sources de 2 à 4 ou de 4 à 8. Dans l'implantation actuelle du système, nous avons fixé le nombre de sources à 7 comme expliqué à la section 4.6.

Dans la figure 4.17, la quantification des coefficients principaux n'a pas été incorporée. La qualité des flux reconstruits et par conséquent des images reconstruites au récepteur dépend énormement du quantificateur utilisé pour les coefficients principaux. Dans la section suivante, nous allons décrire le quantificateur que nous avons retenu et les conséquences de ce choix sur les flux reconstruits.



(a) Séquence "Rubik"

(b) Séquence "Taxi"

(c) Séquence "Nasa"

(d) Séquence "Tunnel"

Figure 4.17 : Effet du nombre de sources utilisées pour la classification de l'amplitude et de la phase sur l'erreur d'angle pour chacune des séquences types de la figure 4.1

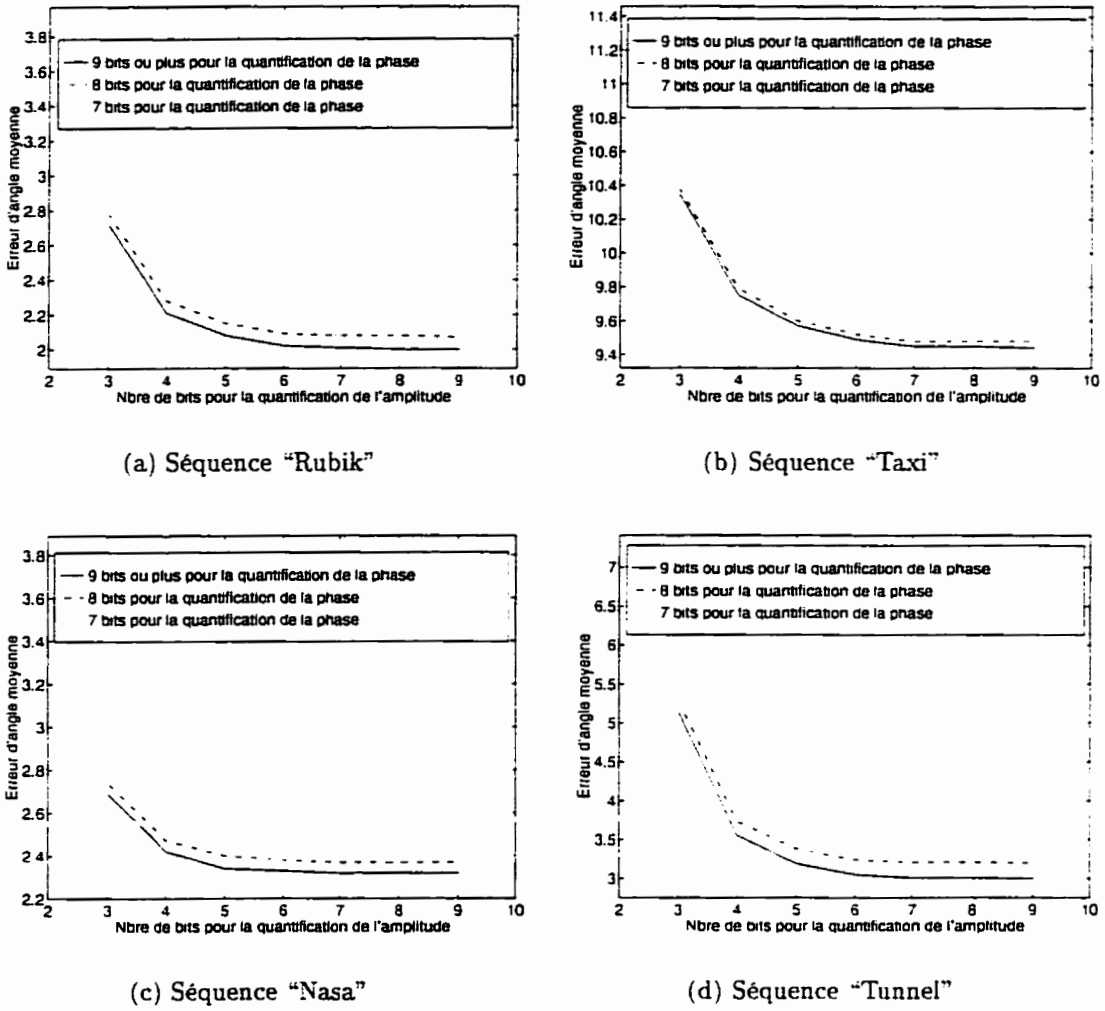
4.3.2.3 Quantification des coefficients principaux

Puisque la distribution des coefficients principaux n'est pas disponible, nous devons les quantifier avec un quantificateur général (qui ne dépend pas du signal à son

entrée). Il n'y a aucune raison pour supposer que certaines valeurs sont plus probables pour les coefficients principaux que d'autres. Un quantificateur uniforme est donc le choix idéal puisqu'il est simple à implanter et optimal pour un signal ayant une distribution uniforme. Le nombre de bits de ce quantificateur est de $\log_2(z+1)$ où z est le nombre de niveaux dans ce dernier.

Notons q_a et q_θ le nombre de bits dans le quantificateur des coefficients principaux de l'amplitude et de la phase respectivement. La figure 4.18 nous montre l'effet du choix de q_a et q_θ sur le flux reconstruit. Puisqu'une erreur sur la phase du flux a plus d'influence sur les images reconstruites qu'une erreur sur son amplitude (Lin et Barron 1994), nous avons décidé d'utiliser plus de résolution dans le quantificateur de la phase que dans celui de l'amplitude. Nous remarquons dans la figure 4.18 que le gain de qualité en utilisant q_θ supérieur à 9 est invisible pour chacune des séquences. La valeur $q_\theta = 9$ est par conséquent utilisée pour le quantificateur de phase. A la figure 4.19, nous avons l'amélioration de qualité en terme de pourcentage à chaque fois qu'on rajoute un bit au quantificateur de l'amplitude pour la valeur choisie de $q_\theta = 9$. Pour la valeur x en abscisse, nous avons en ordonnée $\frac{Err(x-1)-Err(x)}{Err(x-1)}$. Il est évident qu'à chaque fois que nous augmentons q_a, un bit additionnel doit être trannsmis par bloc pour représenter la composante principale de l'amplitude de ce dernier. Nous remarquons à la figure 4.19 que le gain sur la qualité du mouvement en gardant 6 bits au lieu de 5 est de 3%, 1%, 0.5% et 5% pour les quatre séquences respectivement. Dans l'implantation actuelle du système, nous considérons que ce gain ne justifie pas

le rajout de 1 bit/bloc et la valeur $q_a = 5$ est par conséquent utilisée. Cette valeur est tout simplement un choix et n'est bien sûr pas imposée.



(a) Séquence "Rubik"

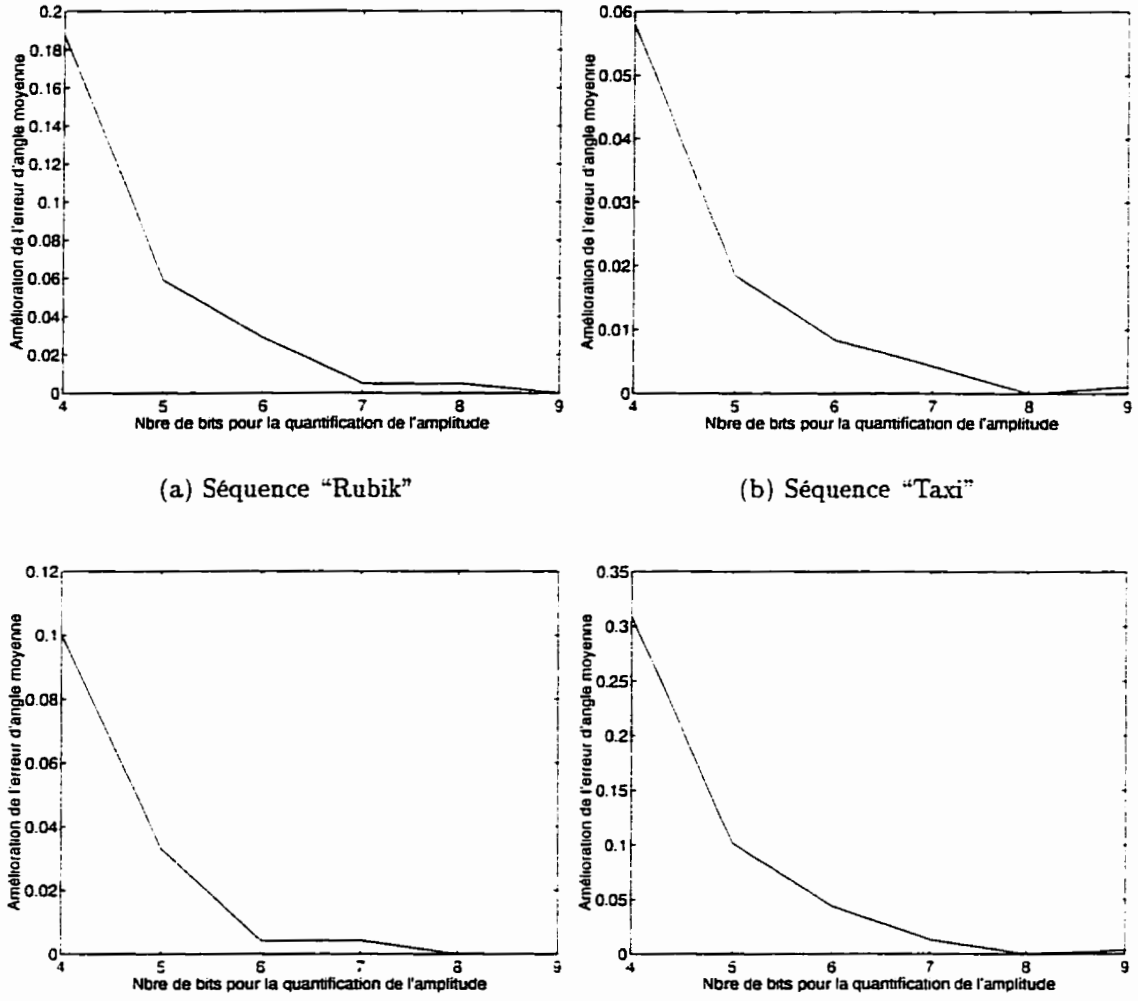
(b) Séquence "Taxi"

(c) Séquence "Nasa"

(d) Séquence "Tunnel"

Figure 4.18 : Effet du nombre de bits gardés pour la quantification du coefficient principal de l'amplitude et de la phase sur l'erreur d'angle pour chacune des séquences types de la figure 4.1

Il est évident que pour reconstruire les flux au récepteur, ce dernier doit utiliser le même quantificateur que l'émetteur. Les valeurs de $\max y^a(1)$, $\min y^a(1)$, $\max y^\theta(1)$



(a) Séquence "Rubik"

(b) Séquence "Taxi"

(c) Séquence "Nasa"

(d) Séquence "Tunnel"

Figure 4.19 : Amélioration de qualité en terme de pourcentage à chaque fois qu'on rajoute un bit au quantificateur de l'amplitude pour la valeur choisie de $q_\theta = 9$

et $\min y^\theta(1)$ doivent par conséquent être transmise au récepteur pour chaque nouvelle image. Nous transmettons la variation de ces dernières dans le temps car ces variations sont normalement petites et peuvent par conséquent être codées avec moins de bits. Nous allons noter ces variations par $\Delta \max y^a(1)$, $\Delta \min y^a(1)$, $\Delta \max y^\theta(1)$ et

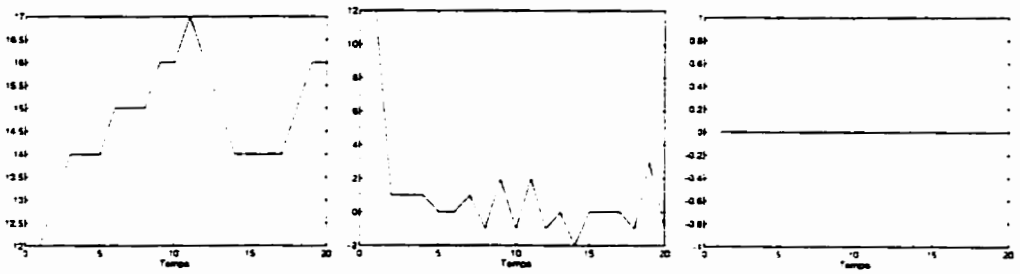
$\Delta \min y^{\theta}(1)$ respectivement.

Par exemple, à la figure 4.20 nous avons la variation de $\max y^{a}(1)$, $\min y^{a}(1)$, $\max y^{\theta}(1)$ et $\min y^{\theta}(1)$ et de $\Delta \max y^{a}(1)$, $\Delta \min y^{a}(1)$, $\Delta \max y^{\theta}(1)$ et $\Delta \min y^{\theta}(1)$ dans le temps pour la séquence "Rubik". Pendant que $\max y^{a}(1)$, $\min y^{\theta}(1)$ et $\max y^{\theta}(1)$ varient dans les domaines $[13, 17]$, $[-73, -58]$ et $[57, 72]$ respectivement et doivent être représentés par 7^{5}, 9 et 9 bits respectivement, $\Delta \max y^{a}(1)$, $\Delta \min y^{\theta}(1)$ et $\Delta \max y^{\theta}(1)$ varient tous dans le domaine $[-2, 3]$ et peuvent être représentés avec 3 bits chacun d'où un gain de compression. Il est évident que la transmission de $\Delta \max y^{a}(1)$, $\Delta \min y^{a}(1)$, $\Delta \max y^{\theta}(1)$ et $\Delta \min y^{\theta}(1)$ ne permet pas d'adapter les quantificateurs aussi vite que la transmission de $\max y^{a}(1)$, $\min y^{a}(1)$, $\max y^{\theta}(1)$ et $\min y^{\theta}(1)$. En effet, des effets d'escalier peuvent être introduits si les limites des quantificateurs changent plus vite que les changements permis. Par exemple, si $\Delta \max y^{a}(1)$ est représenté avec 3 bits, un changement supérieur à 3 de $\max y^{a}(1)$ ne pourra pas être transmis au récepteur et sera effectué sur plusieurs itérations d'où un retard dans l'adaptation comme nous pouvons le voir à la figure 4.21.

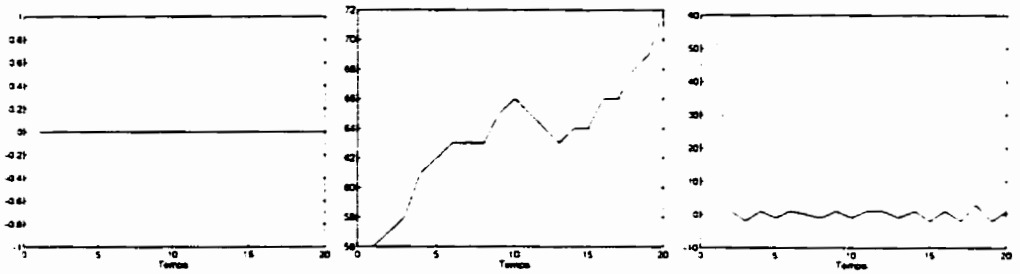
4.4 Reconstruction des images

Pour que notre système de compensation de mouvement nous offre une bonne compression tout en gardant une bonne qualité des images, il faut s'assurer que la

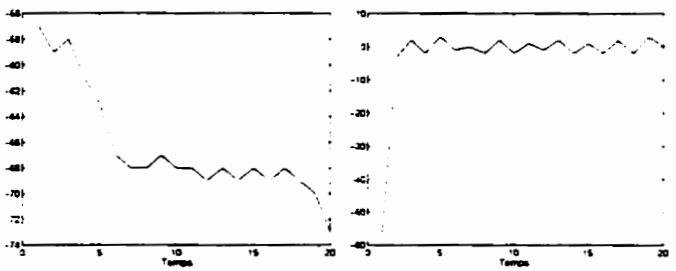
[5] 1 bit pour le signe et 6 bits pour représenter la valeur ($\lceil \log_2(17) \rceil = 6$, où $\lceil x \rceil$ est le plus petit entier supérieur à x)



(a) $\max y^a(1)$ (b) $\Delta \max y^a(1)$ (c) $\min y^a(1)$



(d) $\Delta \min y^a(1)$ (e) $\max y^\theta(1)$ (f) $\Delta \max y^\theta(1)$



(g) $\min y^\theta(1)$ (h) $\Delta \min y^\theta(1)$

Figure 4.20 : Variation de $\max y^a(1)$, $\min y^a(1)$, $\max y^\theta(1)$ et $\min y^\theta(1)$ et de $\Delta \max y^a(1)$, $\Delta \min y^a(1)$, $\Delta \max y^\theta(1)$ et $\Delta \min y^\theta(1)$ dans le temps pour la séquence "Rubik"

reconstruction des images au récepteur ne cause pas d'erreur excessive.

La reconstruction peut se faire de deux façons: inverse ou directe (Lin et Barron

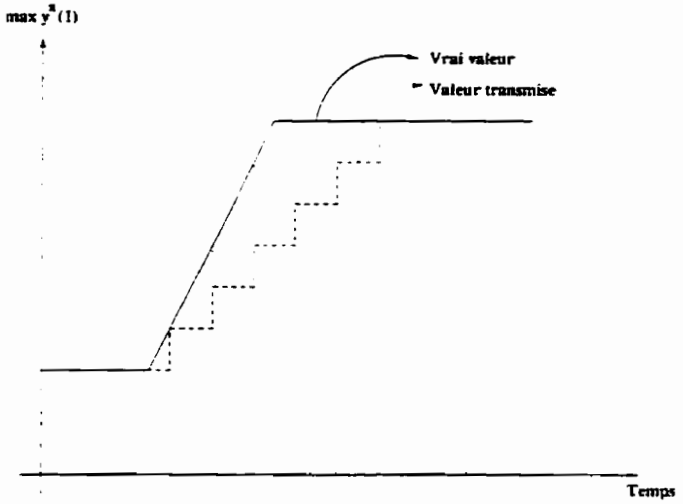


Figure 4.21 : Effet d'escalier sur l'adaptation des quantificateurs causé par un grand changement dans les limites de ces derniers

1994).

4.4.1 Reconstruction inverse

Si le vecteur de mouvement calculé entre l'image au temps t et celle au temps $t+1$ pour le pixel (x, y) est $(v_{1_t}(x, y), v_{2_t}(x, y))$, nous pouvons reconstruire l'intensité de ce pixel au temps $t+1$ en nous basons sur l'équation:

$$\hat{I}_{t+1}(x, y) = I_t(x - v_{1_t}(x, y), y - v_{2_t}(x, y)) \tag{4.22}$$

La valeur d'intensité du sous-pixel $I_t(x - v_{1_t}(x, y), y - v_{2_t}(x, y))$ doit être interpolée à partir des pixels voisins (figure 4.22). Nous utilisons une des méthodes suivantes pour cette interpolation:

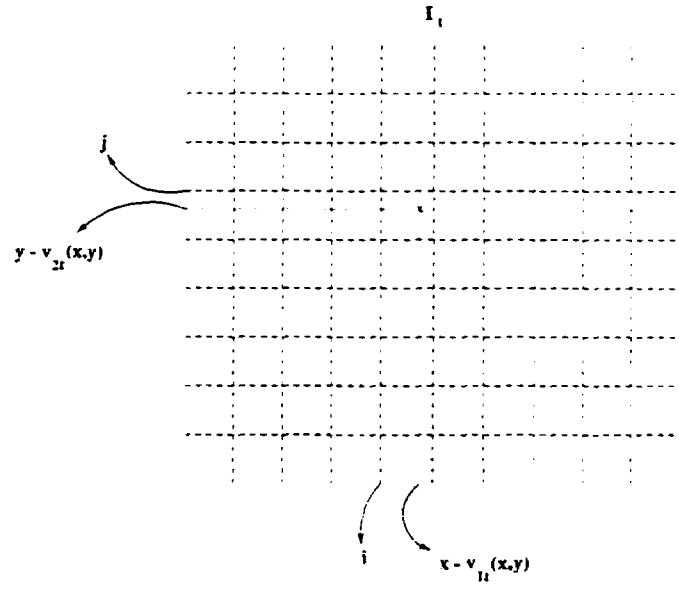


Figure 4.22 : Reconstruction inverse

1. **Interpolation bilinéaire** Nous utilisons les 4 pixels (i,j), $(i+1,j)$, $(i,j+1)$, $(i+1,j+1)$ qui entourent le sous-pixel $(x-v_{1_t}, y-v_{2_t})$ pour interpoler l'intensité de ce sous-pixel selon l'équation:

$$I_t(x-v_{1_t}, y-v_{2_t}) = (1-t)(1-u)I_t(i,j) + t\,(1-u)I_t(i+1,j) + t\,u\,I_t(i+1,j+1) + (1-t)\,u\,I_t(i,j+1) \quad (4.23)$$

où $t = x - v_{1_t} - i$ et $u = y - v_{2_t} - j$. Cette méthode est rapide mais cause beaucoup d'erreurs aux occlusions (Lin et Barron 1994).

2. **Interpolation par polynômes en 2 dimensions** Un voisinage de $n \times n$ est utilisé pour interpoler l'intensité du sous-pixel $I_t(x - v_{1_t}(x,y), y - v_{2_t}(x,y))$. Nous effectuons deux interpolations en une dimension pour x et y respective-

ment. Chacune des lignes du voisinage $(j - n/2, \ldots, j + n/2)$ est utilisée pour reconstruire l'intensité au pixel de cette ligne ayant comme abcisse $x - v_{1_t}(x, y)$. Par la suite, ces n valeurs sont utilisées pour reconstruire l'intensité du sous-pixel $I_t(x - v_{1_t}(x, y), y - v_{2_t}(x, y))$ en faisant l'interpolation en y. Des polynômes de degré $(n - 1)$ sont utilisés pour chacune des interpolations en une dimension. Les n points utilisés pour effectuer l'interpolation sont suffisants pour estimer les coefficients de ces polynômes. Quand le degré des polynômes augmente, on risque d'obtenir des résultats non désirés à cause de l'oscillation de ces derniers (Johnson et Riess 1982).

3. **Interpolation par splines cubiques en 2 dimensions** Comme dans la méthode précédente, nous effectuons deux interpolations en une dimension pour x et y respectivement. Pour chaque dimension, nous interpolons avec des splines cubiques $g(x)$ passant par les n points du voisinage. Nous avons donc les propriétés suivantes:

   (a) $g(x)$ est un polynôme de degré 3 entre chaque couple de 2 points;

   (b) $g(x)$, $g'(x)$ et $g''(x)$ sont continues;

   (c) $g(x)$ passe par les n points.

   La première propriété nous permet d'éviter les problèmes d'oscillation des polynômes ce qui nous permet d'effectuer une meilleure interpolation.

4.4.2 Reconstruction directe

Nous nous basons sur l'équation:

$$\hat{I}_{t+1}\left(x + v_{1_t}(x,y), y + v_{2_t}(x,y)\right) = I_t(x,y) \tag{4.24}$$

Contrairement à la reconstruction inverse (équation 4.22), l'interpolation est effectuée sur l'image $t+1$. Les valeurs d'intensité de l'image t sont déplacées aux sous-pixels de l'image $t+1$ et il faut interpoler ces derniers pour trouver les intensités lumineuses sur la grille image. Cette interpolation peut être effectuée à partir des intensités ou des déplacements:

1. **Interpolation directe basée sur les intensités** Les intensités lumineuses des quatre sous-pixels (x_1, y_1), (x_2, y_2),(x_3, y_3) et (x_4, y_4) les plus proches du pixel (x, y) dans les quatre quadrants d'un système de coordonnées placé sur ce pixel sont utilisées pour interpoler l'intensité lumineuse de ce pixel (voir la figure 4.23). Nous commençons par construire la droite $(a_1, b_1)-(a_2, b_2)$ parallèle à $(x_1, y_1)-(x_2, y_2)$ et passant par (x, y). Une interpolation linéaire nous permet d'interpoler l'intensité fictive I_5 du sous-pixel (a_1, b_1) entre I_2 et I_3. De même, I_6 est interpolée entre I_1 et I_4. Enfin, nous pouvons interpoler l'intensité I du pixel (x, y) entre I_5 et I_6.

   Cette procédure peut être répétée en construisant des droites parallèles à $(x_2, y_2)-(x_3, y_3)$, à $(x_3, y_3)-(x_4, y_4)$ et à $(x_4, y_4)-(x_1, y_1)$. La moyenne des quatre valeurs

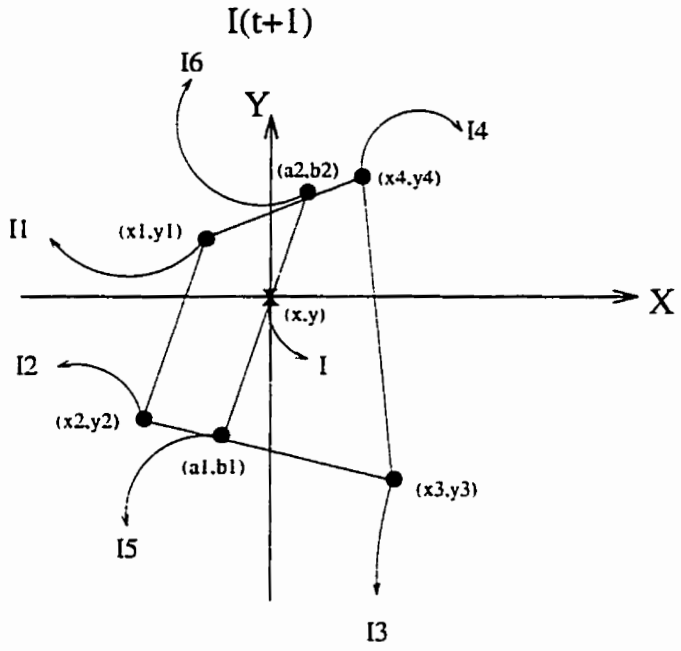


Figure 4.23 : Reconstruction directe

trouvées de I sera choisie pour l'intensité du pixel (x, y).

2. **Interpolation directe des déplacements** Au lieu d'interpoler les intensités lumineuses des quatre sous-pixels (x_1, y_1), (x_2, y_2), (x_3, y_3) et (x_4, y_4), nous pouvons interpoler leurs vecteurs de déplacement vers l'image précédente pour obtenir un vecteur de déplacement du pixel (x, y) de l'image $t + 1$ vers un sous-pixel de l'image t. Sur ce sous-pixel, nous effectuons une interpolation inverse comme expliqué à la section 4.4.1.

   Cette méthode donne des meilleurs résultats car les vitesses varient de façon plus douce que les intensités et leur erreur d'interpolation est par conséquent plus petite. Nous avons implanté cette méthode dans notre système pour la reconstruction des images. Comme proposé par Lin et Barron (1994), nous avons

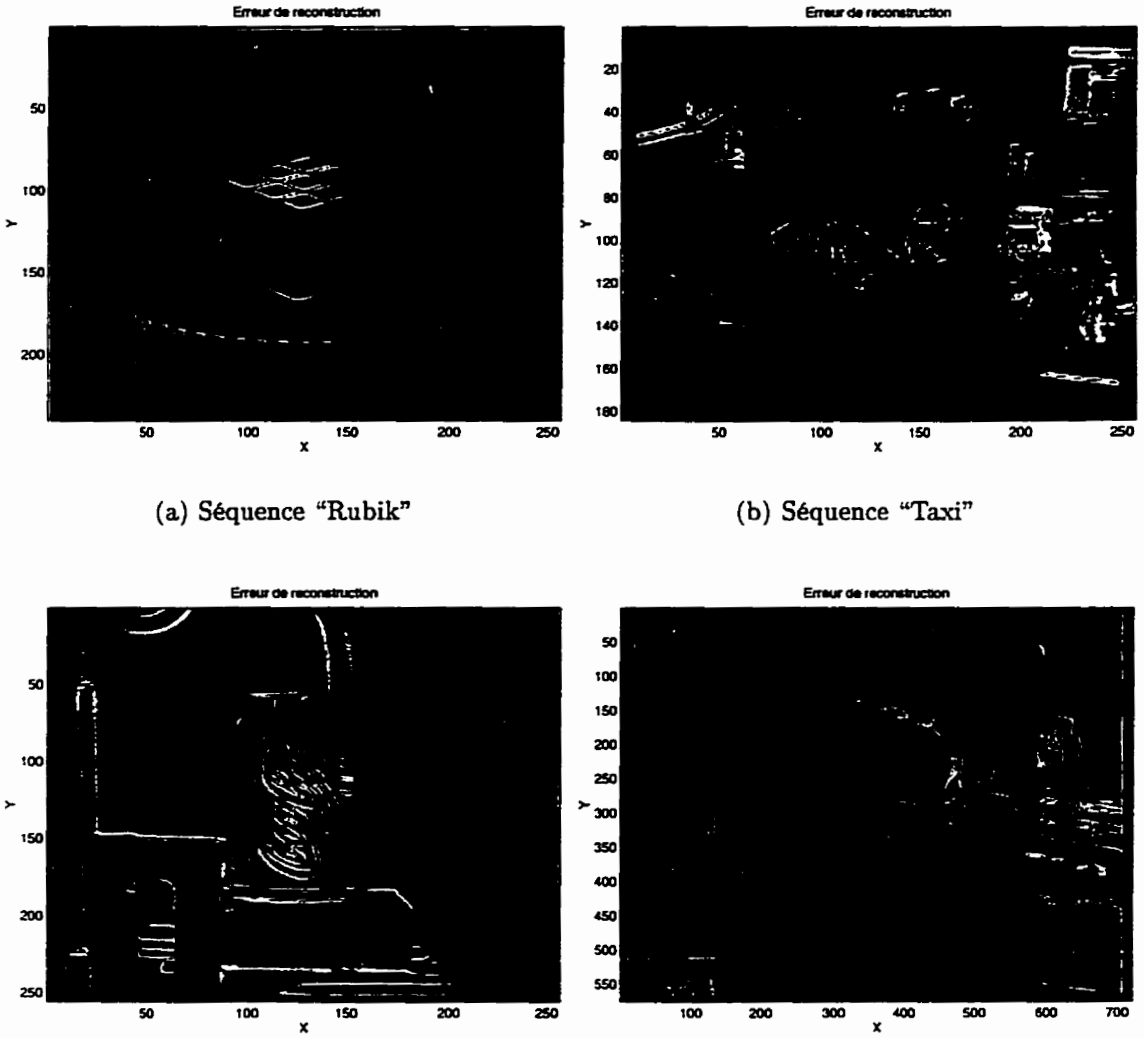
filtré le flux en utilisant un filtre Gaussien d'écart type 1.5 avant d'effectuer la reconstruction. A la figure 4.24, nous montrons l'erreur de reconstruction obtenue pour chacune des séquences de test de la figure 4.1 en se basant sur le flux original (non compressé) pour faire la reconstruction (l'erreur de reconstruction est représentée en niveaux de gris, une luminance élevée correspondant à une grande erreur). L'erreur quadratique moyenne sur ces images est donnée au tableau 4.3. Nous remarquons que l'erreur de reconstruction n'est pas nulle, qu'elle est concentrée aux occlusions et qu'elle s'accumule. Pour pallier à ce problème, nous devons transmettre une image de mise à jour à chaque fois que cette erreur devient trop grande.

Tableau 4.3 : Erreur quadratique moyenne en se basant sur le flux original pour faire la reconstruction

Séquence	EQM
"Rubik"	3.48
"Taxi"	5.16
"Nasa"	4.08
"Tunnel"	6.66

4.5 Transmission des images de mise à jour

Définition 4.3 *Les* **images de mise à jour** *sont des images codées spatialement et qui ont pour but de remettre le système à jour et d'empêcher l'accumulation de l'erreur dans le temps.*



(a) Séquence "Rubik"

(b) Séquence "Taxi"

(c) Séquence "Nasa"

(d) Séquence "Tunnel"

Figure 4.24 : Erreur de reconstruction obtenue pour chacune des séquences types de la figure 4.1 en se basant sur le flux original pour faire la reconstruction

Ces images ont les trois objectifs suivants:

1. empêcher l'accumulation de l'erreur dans le temps;

2. remettre le système à jour en cas de coupure de scène;

3. remettre les matrices de transformation à jour.

Nous transmettons ces images à chaque fois que l'erreur quadratique moyenne sur les blocs dynamiques dépasse un seuil. Ce seuil règle le rapport taux/distorsion du système. En effet, plus ce seuil est grand, plus l'erreur tolérée est grande au gain d'une plus grande compression. Dans les résultats expérimentaux que nous présentons au chapitre 5, plusieurs valeurs seront utilisées pour ce seuil selon le rapport de compression désiré.

Comme dans les images codée par compensation de mouvement, les blocs statiques gardent les mêmes intensités que l'image précédente. D'autre part, les blocs dynamiques sont codés par TCD. Les figures 4.25 et 4.26 nous montrent l'erreur quadratique moyenne sur les blocs dynamiques et sur toute l'image respectivement selon le nombre de coefficients que nous gardons par bloc. De plus, la figure 4.27 montre l'amélioration de qualité en terme de pourcentage à chaque fois que nous rajoutons un coefficient à transmettre de la TCD des images de mise à jour. Pour la valeur x en abscisse, nous avons en ordonnée $\frac{EQM(x-1)-EQM(x)}{EQM(x-1)}$. Dans l'implantation actuelle, nous considérons que l'ajout d'un coefficient est justifiable s'il introduit une amélioration d'au moins 10% sur les blocs dynamiques. Nous remarquons à la figure 4.27 que ce seuil correspond à garder 15 coefficients par bloc. Le seuil de 10% est tout simplement un choix et dépend surtout de l'application en cours. En effet, pour les applications où nous désirons une grande qualité des images au dépend d'une compression faible, nous tolérons un grand nombre de coefficients à transmettre. Par

contre. pour les applications à bas débit, ce nombre doit être faible.

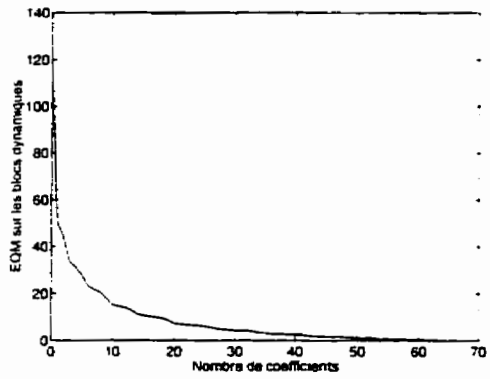
4.6 Implantation

Pour initialiser le système. la transmission de deux images $I(0)$ et $I(1)$ est nécessaire pour initialiser la représentation du mouvement à celui calculé entre ces deux images. Ces deux images sont codées dans le domaine spatial. En d'autres mots. elles sont des images de mise à jour dont la première est entièrement codée et seuls les blocs dynamiques de la deuxième par rapport à la première sont transmis. A l'émetteur comme au récepteur le flux optique aux blocs dynamiques est calculé entre ces deux images et utilisé pour initialiser les matrices de transformation et les limites des quantificateurs ($\max y^a(1)$,$\min y^a(1)$,$\max y^\theta(1)$ et $\min y^\theta(1)$).

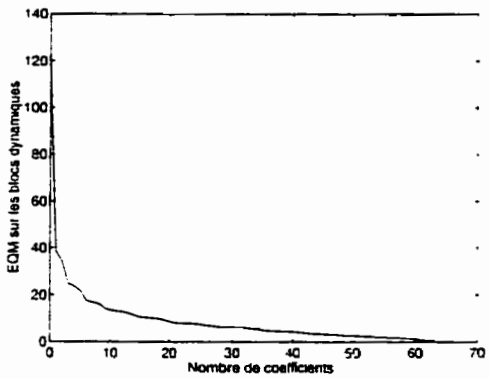
Par la suite, les informations suivantes doivent être transmises pour chacune des images:

1. **type de l'image:** pour indiquer si c'est une image codée par compensation de mouvement (de type P) ou une image de mise à jour (de type I). Ceci peut être représenté avec un bit qui est égal à 1 pour les images de type P et à 0 pour les images de type I.

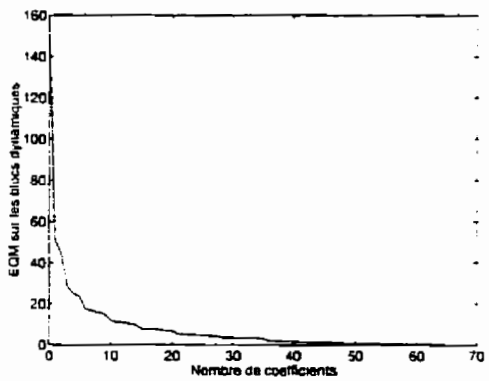
2. **masque des blocs dynamiques:** pour indiquer la position des blocs dynamiques. Ceci peut être représenté avec un bit par bloc. Un bit égal à 1 signifie un bloc dynamique et un bit égal à 0 signifie un bloc statique. Un



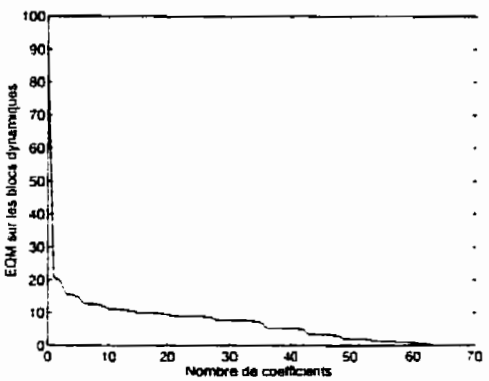
(a) Séquence "Rubik"



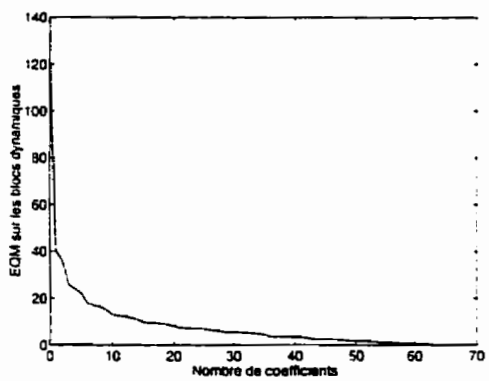
(b) Séquence "Taxi"



(c) Séquence "Nasa"

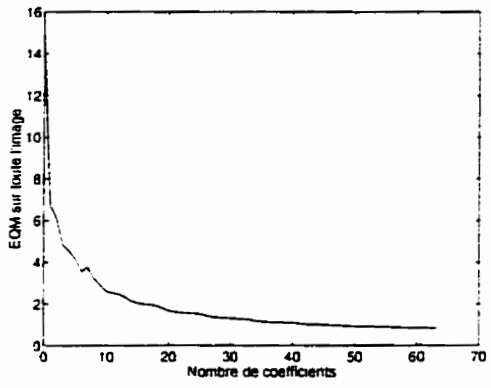


(d) Séquence "Tunnel"

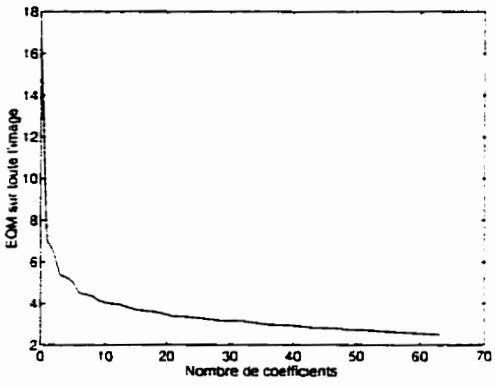


(e) Moyenne sur les quatre séquences

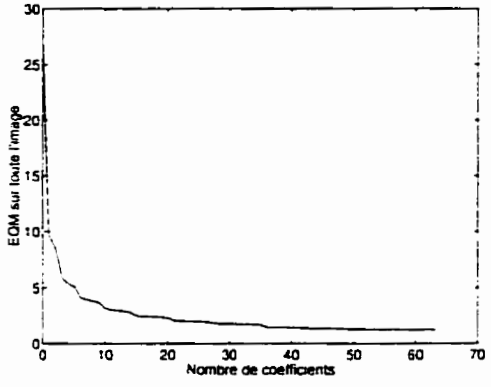
Figure 4.25 : Effet du nombre de coefficients gardés de la TCD sur la moyenne de l'erreur quadratique moyenne sur les blocs dynamiques des images de mise à jour



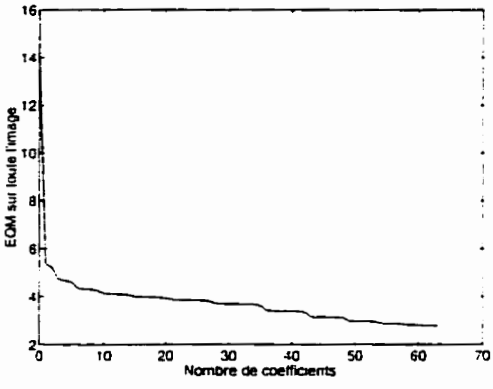
(a) Séquence "Rubik"



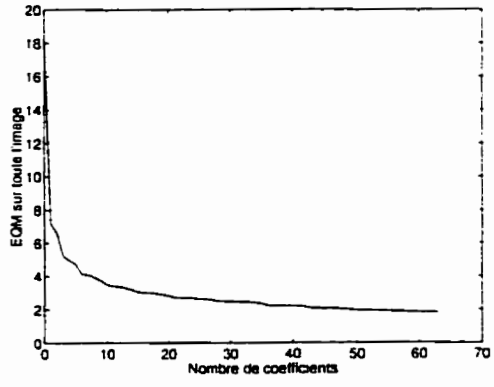
(b) Séquence "Taxi"



(c) Séquence "Nasa"

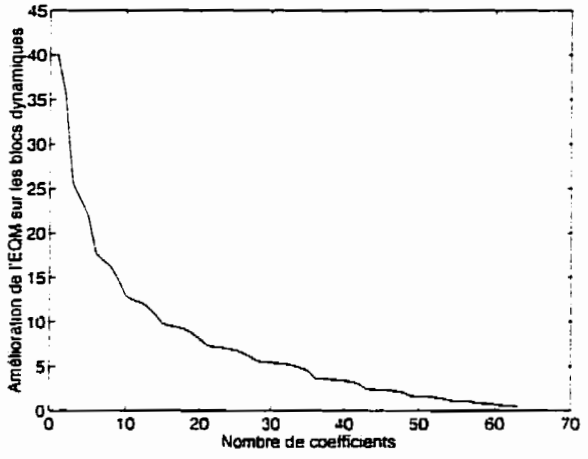


(d) Séquence "Tunnel"

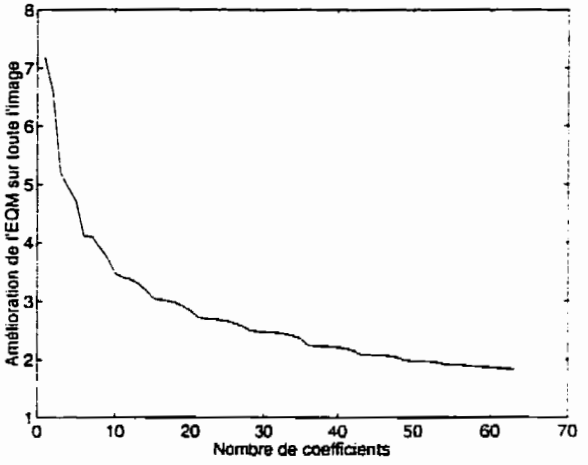


(e) Moyenne sur les quatre séquences

Figure 4.26 : Effet du nombre de coefficients gardés de la TCD sur la moyenne de l'erreur quadratique moyenne sur toute l'image de mise à jour



(a) Sur les blocs dynamiques



(b) Sur toute l'image

Figure 4.27 : Amélioration de qualité en terme de pourcentage à chaque fois qu'on rajoute un coefficient à transmettre de la TCD calculée sur la moyenne des quatre séquences

exemple d'un tel masque est donné à la figure 4.28.

3. **adaptation des quantificateurs:** pour les images de type P comme expliqué

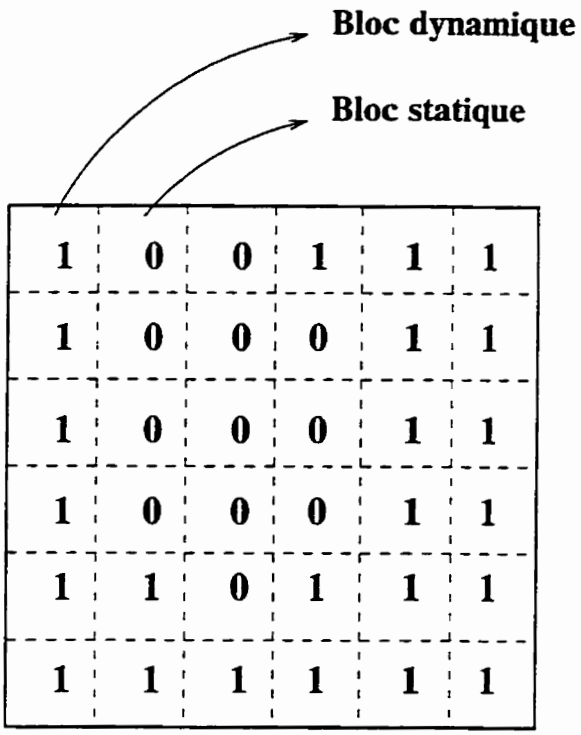


Figure 4.28 : Exemple d'un masque des blocs dynamiques

à la section 4.3.2.3.

4. **information pour les blocs dynamiques:**

   - pour les images de type P, nous transmettons la classification et le coefficient principal de l'amplitude et de la phase:
   - pour les images de type I, nous transmettons les coefficients de la TCD dans le domaine spatial.

En se basant sur les notations du tableau 4.4, le flux binaire transmis est montré à la figure 4.30. Il est évident qu'une erreur sur le type de l'image ou sur le masque des blocs dynamiques peut entraîner la perte de la synchronisation et par conséquent une

Tableau 4.4 : Notations utilisées pour le calcul du rapport de compression

Symbole	Signification
$MAX_x \times MAX_y$	Taille des images
T	Nombre d'images
N_B	Nombre de blocs ($N_B = N_{Bx} \times N_{By} = \frac{MAX_x MAX_y}{N}$ pour des blocs de $\sqrt{N} \times \sqrt{N}$)
L_m	Taille du masque des blocs dynamiques
N_c	Nombre de sources utilisées pour la classification
L_c	Nombre de bits utilisés pour la classification ($L_c = log_2 N_c$)
P_{mj}	Pourcentage des images de mise à jour dans la séquence
$P_d(t)$	Pourcentage des blocs dynamiques dans l'image au temps t
N_{mj}	Nombre de coefficients gardés de la DCT par bloc dynamique pour les images de mise à jour
L_{mj}	Nombre de bits gardés de la DCT par bloc dynamique pour les images de mise à jour
q_a	Nombre de bits utilisés pour la quantification du coefficient principal de l'amplitude
q_θ	Nombre de bits utilisés pour la quantification du coefficient principal de la phase
L_q	Nombre de bits utilisés pour l'adaptation des quantificateurs ($L_q = log_2 \Delta_q$)

accumulation non détectable des erreurs. Pour pallier à ce problème, nous protégeons ces deux informations avec des codes de parité. De plus, un signal de synchronisation est transmis à la fin de chaque image.

4.6.1 Protection du type d'image et du masque des blocs dynamiques

Nous avons représenté le type de l'image avec cinq bits au lieu d'un seul. Le même bit est donc répété cinq fois. Au récepteur, le type de l'image est choisi selon le bit répété au moins trois fois sur les cinq bits. Dans les situations où la décision précédente est erronée, la synchronisation nous permettra de détecter cette erreur et de la corriger comme expliqué à la section suivante.

Le masque des blocs dynamiques est muni d'un code de parité. Un exemple est montré à la figure 4.29. Nous rajoutons au masque une ligne et une colonne représentant les bits de parité. Les valeurs à la ligne inférieure correspondent à 0 si le nombre de 1 dans leur colonne correspondante est pair et à 1 si ce nombre est impair. De même, les bits de la colonne à droite correspondent à la parité de leur ligne respective. Ce code nous permet, pour sûr, de corriger une erreur dans le masque et d'en détecter trois. Selon la localisation de ces erreurs, il est possible d'en détecter et d'en corriger plus. Transmettre ce code de parité augmente la taille du masque L_m de N_B à $N_B + log_2 N_{Bx} + log_2 N_{By}$ bits/image.

4.6.2 Synchronisation des images

Il s'agit de transmettre un code de synchronisation à la fin de chaque image avant de transmettre le type de l'image suivante. Nous devons nous assurer qu'aucune combinaison des informations à transmettre nous donne ce code par erreur.

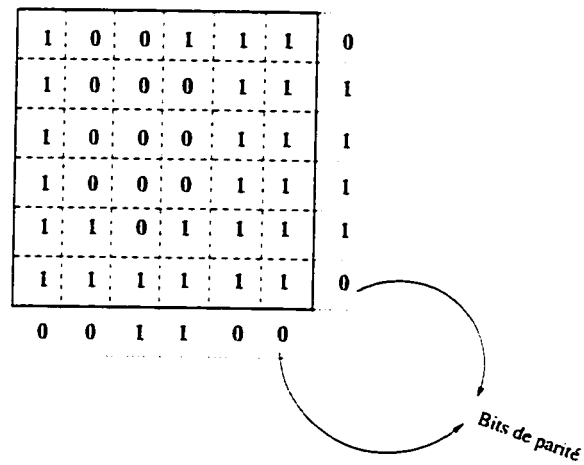


Figure 4.29 : Protection du masque des blocs dynamiques de la figure 4.28 par un code de parité

Pour ce faire, il suffit de garder un mot libre (une série de x "1" de suite (1^x) par exemple) qui n'est alloué à aucune des valeurs possibles des symboles transmis (Δ_q, C_a, C_θ, $y^a(1)$, $y^\theta(1)$ et tous les coefficients TCD). Ceci nous garantit que la plus grande série de 1 possible est celle de deux symboles successifs et donc de pouvoir définir une série de 1 de longueur suffisemment grande pour ne jamais apparaitre dans les données transmise et de s'en servir de synchronisation.

Un symbole qui peut prendre $d = 2^x$ valeurs possibles et qui peut donc être décrit avec x bits, doit être décrit avec $x + 1$ bits. D'autre part, si un symbole peut prendre $2^x < d < 2^{x+1}$ valeur possibles, il pourra être décrit avec x bits sans avoir à allouer le mot 1^x à aucune valeur. Ceci peut être écrit en forme mathématique:

$$\text{Nombre de bits alloués} = \lfloor log_2 d \rfloor + 1 \tag{4.25}$$

où $\lfloor x \rfloor$ signifie le plus grand entier inférieur à x.

Exemple 4.1 *Nous utilisons 7 sources de l'amplitude et de la phase pour faire la classification et donc le code* 1^3 *n'est jamais présent pour la classification. D'autre part, le nombre de pas dans les quantificateurs des coefficients principaux de l'amplitude et de la phase est fixé à* $2^5 - 1$ *et* $2^9 - 1$ *respectivement et donc 5 et 9 bits sont suffisant pour représenter les coefficients principaux de l'amplitude et de la phase respectivement. Enfin, les coefficients de la TCD sont quantifiés à* $2^8 - 1$ *bits et donc ces coefficients peuvent être décrits avec 8 bits. La plus grande série de 1 possible peut provenir soit de la suite* $C_\theta\ y^\theta(1)$ *qui peut nous donner au maximum* 0111^80 *ou de deux TCD de suite qui peut nous donner au maximum* $01^7\ 1^70$. *La plus grande série de 1 possible est donc de 14 et par conséquent, le mot de code* 1^{15} *peut servir de synchronisation.*

Nous allons noter le nombre de bits de synchronisation par L_s.

L'algorithme à suivre au décodeur est le suivant:

- lire les bits du type d'image et décider sur ce dernier:
- lire le masque des blocs dynamiques, corriger les erreurs corrigible et détecter s'il y en a qui ne le sont pas;
- sachant que l'information de mouvement et de TCD est de taille fixe par bloc dynamique, calculer le nombre de bits total représentant l'image et vérifier si la synchronisation est présente à la fin de ce code;
- si la synchronisation n'y est pas, vérifier avec l'autre type d'image;

- si on ne trouve pas la synchronisation avec aucun des types, nous avons une erreur dans le masque ou dans la synchronisation.
  - chercher la synchronisation:
  - décoder l'information avant la synchronisation trouvée en supposant qu'une synchronisation manque à cause d'une erreur sur cette dernière.

Il est évident que le système est sensible aux erreurs dans le masque. Ces erreurs ne seront corrigées qu'à la prochaine image de mise à jour (sauf si le décodeur peut communiquer avec l'émetteur pour lui indiquer l'erreur). Le fait d'avoir rajouté les codes de parité diminue largement la probabilité de ces erreurs[6].

Le rapport de compression obtenu peut être exprimé selon la relation:

$$\frac{T \times MAX_x MAX_y \times 8\text{bits}}{\underbrace{T(L_m+L_s+5)}_{A}+\sum_{t=1}^{T}\underbrace{\{P_{mj}(P_d(t)N_B L_{mj})}_{B}+\underbrace{(1-P_{mj})[L_q+(2L_c+q_a+q_\theta)P_d(t)N_B]\}}_{C}} : 1 \tag{4.26}$$

Dans l'équation précédente, le terme A correspond au nombre de bits nécessaires pour la détermination des images de mise à jour (5 bits/image), la transmission du masque des blocs dynamiques (L_m bits/image) et la transmission de la synchronisation (L_s bits/image). D'autre part, le terme B correspond au nombre de bits transmis pour les coefficients gardés de la TCD pour les images de type I. Enfin, le terme C correspond au nombre de bits transmis pour la classification, les coefficients prin-

[6]Pour qu'une erreur soit non corrigeable par le code de parité, il faut avoir, au moins, deux erreurs colinéaire dans le masque

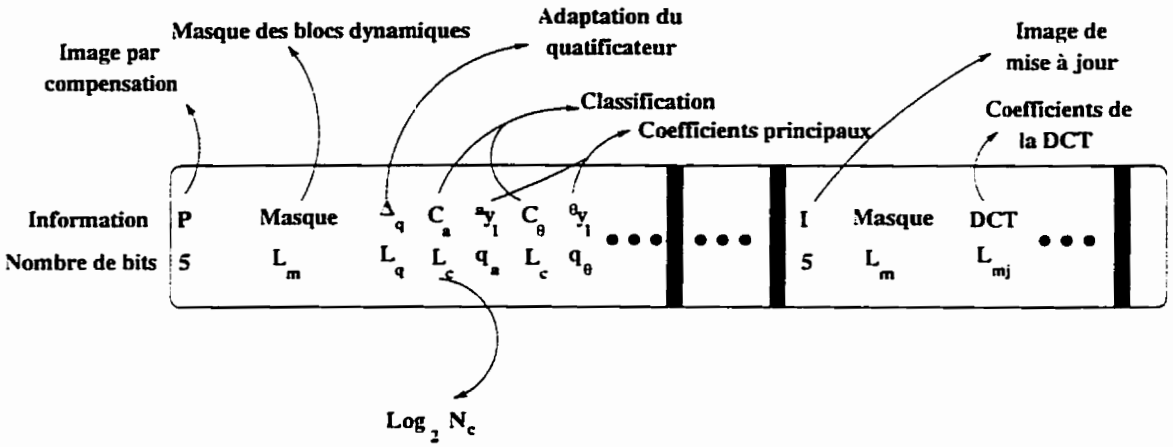


Figure 4.30 : Flux binaire transmis au canal

cipaux des blocs dynamiques et l'adaptation des quantificateurs pour les images de type P.

Il est à noter que l'équation 4.26 ne tient pas compte de l'ajout de compression possible en incorporant le codage entropique avant la transmission.

Exemple 4.2 *Si nous codons* $T = 70$ *images de la séquence "Rubik"* ($MAX_x \times MAX_y = 256 \times 240$) *et nous supposons que* $P_d = 15\%$ *des blocs sont dynamiques et que nous transmettons une image de mise à jour à toutes les* 20 *images* ($P_{mj} = \frac{1}{20} = 5\%$) *codée en gardant* 15 *coefficients de la TCD dans les blocs dynamiques. nous aurons un rapport de compression de* 100 : 1.

Chapitre 5

Résultats expérimentaux

Au chapitre 4, nous avons présenté tous les modules de notre système. Des tests sur ces modules nous ont permis de fixer les paramètres qui y sont associés. Les valeurs de ces paramètres sont données au tableau 5.1. Seul le seuil de l'erreur quadratique moyenne à partir duquel on transmet les images de mise à jour S_{EQM} n'est pas fixé. En effet, c'est ce seuil qui nous permet de contrôler le taux de compression obtenu avec notre système en variant le pourcentage des images de mise à jour P_{mj}. Nous avons testé notre système sur les séquences de la figure 4.1 avec plusieurs valeurs de ce seuil. Le rapport signal à bruit crête à crête (RSBC) a été utilisé comme paramètre de qualité des images reconstruites au récepteur. Le RSBC s'exprime en décibels (dB). A titre indicatif, un rapport $RSBC = 30dB$ correspond à une erreur $EQM = 8$. Inversement à l'EQM qui est une mesure d'erreur, le RSBC est une mesure de qualité et donc plus la valeur du RSBC est élevé, meilleure est la qualité de l'image. Le

RSBC est défini par l'équation:

$$RSBC = 10 \times log_{10}(\frac{255^2}{EQM^2}) \tag{5.1}$$

Aux tableaux 5.2, 5.3, 5.4 et 5.5, nous avons la liste de tous les tests effectués pour les séquences "Rubik", "Taxi", "Nasa" et "Tunnel" respectivement. Le RSBC dans le temps obtenu avec chacun des ces tests est montré à la figure 5.1.

Tableau 5.1 : Notations utilisées pour le calcul du rapport de compression

Paramètre	Valeur
Nombre de sources	7
Nombre de bits pour la quantification du coefficient principal de l'amplitude	5
Nombre de bits pour la quantification du coefficient principal de la phase	9
Seuil pour écrêter la détection des blocs dynamiques ($EQ_{\max}$)	8×8
Nombre de coefficients à garder de la TCD dans le domaine spatial pour les images de mise à jour	15
Nombre de bits pour l'adaptation des quantificateurs	$4 \times 5 = 20^{\dagger}$
† 5 bits pour le maximum et le minimum de l'amplitude et de la phase	

Tableau 5.2 : Tests effectués sur la séquence "Rubik"

S_{EQM}	P_{mj}	Compression
20	6.7%	92 : 1
17	12%	79 : 1
15	18%	69 : 1

Pour évaluer la performance de notre système, nous l'avons comparé avec une version en logiciel de MPEG I. A la figure 5.1, nous montrons le RSBC dans le temps

Tableau 5.3 : Tests effectués sur la séquence "Taxi"

S_{EQM}	P_{mj}	Compression
20	8.3%	97 : 1
17	10%	93 : 1
15	12%	88 : 1

Tableau 5.4 : Tests effectués sur la séquence "Nasa"

S_{EQM}	P_{mj}	Compression
20	10%	73 : 1
17	13%	67 : 1
15	18%	58 : 1

pour plusieurs rapports de compression obtenue en utilisant notre système comparé au standard MPEG I. Les compressions obtenues avec MPEG I sont données au tableau 5.6. Les valeurs moyennes, minimales et maximales du RSBC sont données aux tableaux 5.7, 5.8, 5.9 et 5.10 respectivement. De plus, on peut voir les images reconstruites ayant une erreur maximale et moyenne pour chacune des séquences aux figures 5.2 à 5.9.

Aucune des séquences de test ne vérifie l'hypothèse de mouvement translationnel fronto-parallèle prise dans MPEG I. Nous devons donc nous attendre à des images

Tableau 5.5 : Tests effectués sur la séquence "Tunnel"

S_{EQM}	P_{mj}	Compression
20	14%	75 : 1
17	18%	69 : 1
15	22%	63 : 1

reconstruites de meilleure qualité et avec plus de compression en effectuant le codage avec notre système par rapport à MPEG I. Les résultats varient selon les séquences.

Le plus grand gain de qualité obtenu est avec les séquences "Taxi" et "Tunnel". En effet, la séquence "Taxi" présente peu de blocs dynamiques et donc, le potentiel de compression, grâce au module d'estimation des blocs dynamiques, est excellent. D'autre part, la séquence "Tunnel" est échantillonnée selon le standard PAL SECAM. Dans ce standard, le nombre d'images par seconde est de 25 au lieu de 30 dans le standard NTSC. Ceci signifie que l'intervalle de temps entre les images de cette séquence est supérieur aux autres séquences et donc le mouvement est plus élevé. Ce mouvement est, par conséquent, plus difficile à estimer par un modèle supposant une translation fronto-parallèle ce qui cause la mauvaise qualité de l'image reconstruite avec MPEG I (figure 5.9(a))

La séquence "Nasa" est une séquence difficile à coder car toute l'image bouge selon un mouvement de convergence ou de divergence. Ceci explique que les compressions obtenues avec notre méthode et avec MPEG I soient faibles. Il reste que notre méthode donne un bien meilleur RSBC avec une meilleure compression à cause de l'utilisation du mouvement dense pour la compensation du mouvement. Ceci est confirmé par les images reconstruites à la figure 5.7.

Il est évident que le rapport signal à bruit crête à crête est un critère qui ne tient pas compte du système visuel humain. En effet, nous pouvons avoir des images ayant une erreur globale plus faible que d'autres mais dont l'erreur est plus dérangeante

Tableau 5.6 : Compression obtenue avec le standard MPEG I pour chacune des séquences de test de la figure 4.1

Séquence	Compression
"Rubik"	66 : 1
"Taxi"	64 : 1
"Nasa"	59 : 1
"Tunnel"	53 : 1

Tableau 5.7 : Valeurs moyennes, minimales et maximales du rapport signal à bruit crête à crête sur la séquence "Rubik"

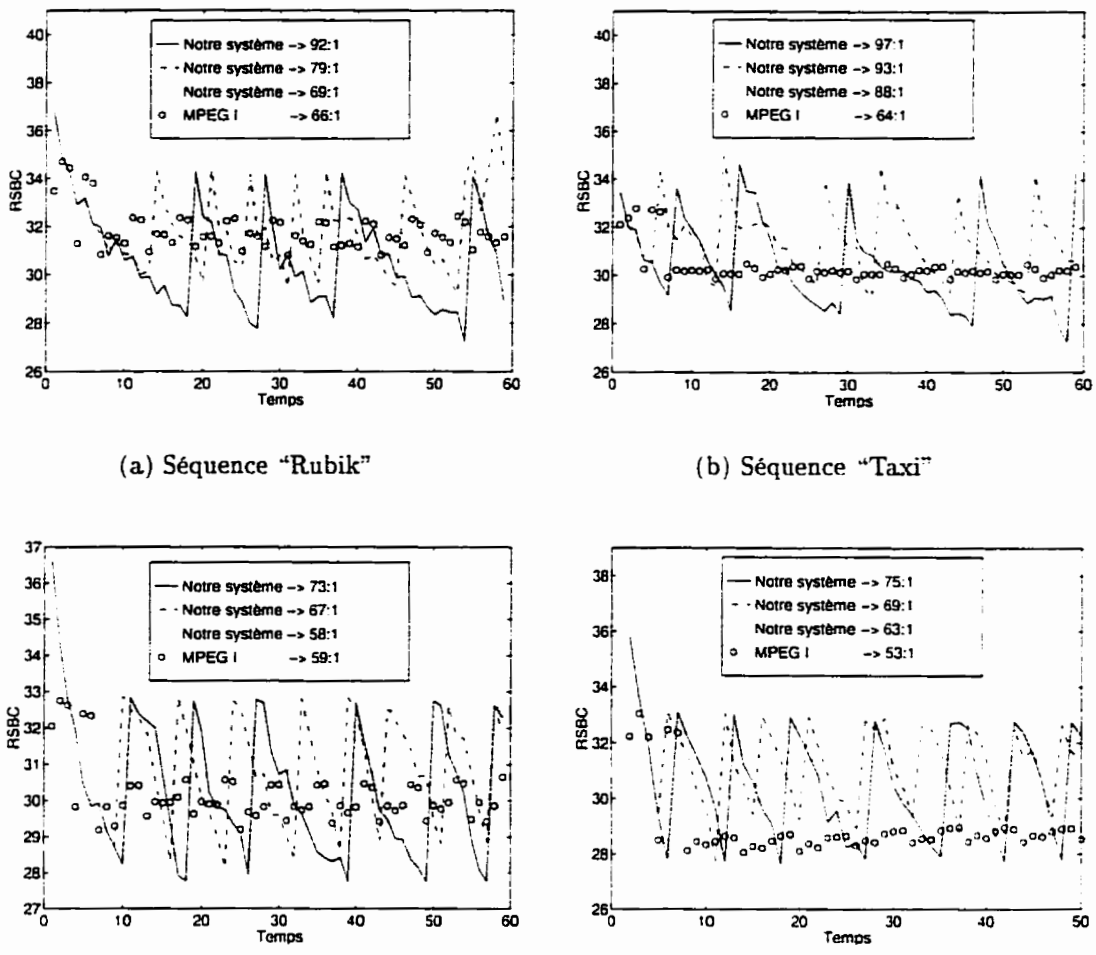
Type de compression	Taux de compression	RSBC moyenne	RSBC minimale	RSBC maximale
MPEG I	66 : 1	31.74	30.81	34.69
Notre système	92 : 1	30.47	27.25	36.84
Notre système	79 : 1	31.81	29.17	36.84
Notre système	69 : 1	32.34	29.91	36.84

pour un observateur humain. Pour confirmer la supériorité de la qualité des images obtenue avec notre système à celles de MPEG I, nous avons réalisé des tests subjectifs sur des sujets choisis au hazard. Nous avons considéré deux sortes de tests subjectifs:

1. Le choix forcé qui consiste à montrer, simultanément, deux séquences à des

Tableau 5.8 : Valeurs moyennes, minimales et maximales du rapport signal à bruit crête à crête sur la séquence "Taxi"

Type de compression	Taux de compression	RSBC moyenne	RSBC minimale	RSBC maximale
MPEG I	64 : 1	30.30	29.87	32.77
Notre système	97 : 1	30.26	27.25	34.67
Notre système	93 : 1	31.30	29.07	34.95
Notre système	88 : 1	31.81	29.15	35.32



(a) Séquence "Rubik"

(b) Séquence "Taxi"

(c) Séquence "Nasa"

(d) Séquence "Tunnel"

Figure 5.1 : Rapport signal à bruit crête à crête en compressant les séquences avec notre système comparée à MPEG I.

observateurs, une reconstruite avec notre système et l'autre avec MPEG I. Les sujets sont contraints à un choix forcé pour dire laquelle ils préfèrent.

2. Une échelle d'évaluation de la dégradation visuelle des images reconstruites avec notre système en 7 points. Cette échelle est représentée sur le tableau 5.11. Nous

Tableau 5.9 : Valeurs moyennes, minimales et maximales du rapport signal à bruit crête à crête sur la séquence "Nasa"

Type de compression	Taux de compression	RSBC moyenne	RSBC minimale	RSBC maximale
MPEG I	59 : 1	30.09	29.19	32.76
Notre système	73 : 1	30.15	27.76	36.63
Notre système	67 : 1	30.81	28.10	36.63
Notre système	58 : 1	31.51	29.58	36.63

Tableau 5.10 : Valeurs moyennes, minimales et maximales du rapport signal à bruit crête à crête sur la séquence "Tunnel"

Type de compression	Taux de compression	RSBC moyenne	RSBC minimale	RSBC maximale
MPEG I	53 : 1	28.89	28.06	33.04
Notre système	75 : 1	30.37	27.65	35.83
Notre système	69 : 1	31.12	27.77	35.83
Notre système	63 : 1	31.64	29.38	35.83

demandons à l'observateur de donner un chiffre entre 1 et 7 de comparaison de qualité entre la séquence qu'il est en train de visualiser et la séquence originale. Cette dernière lui a été présentée au préalable et lui est montrée de nouveau dès qu'il en formule le souhait.

Les tests subjectifs de type choix forcé ont été effectués sur trois sujets. Chacune des trois séquences reconstruites avec notre système (avec trois taux de compressions différents) est comparée avec MPEG I quatre fois pour chacun des sujets.

Les 12 tests précédents ont été présenté de façon aléatoire aux sujets. Le tableau 5.13 récapitule les résultats obtenus. Dans la première colonne, nous avons

Tableau 5.11 : Echelle de visibilité des dégradations

Numérotation	Signification
1	non perceptibles
2	juste perceptibles
3	légères mais définitivement perceptibles
4	non dérangeantes
5	quelque peu dérangeantes
6	définitivement dérangeantes
7	extrèmement dérangeantes

la compression obtenue avec notre système. Les quatre autres colonnes montrent le nombre de fois que chacun des trois sujets a préféré notre système à MPEG I et la moyenne des trois.

Nous remarquons que les séquences reconstruites avec une compression de 92 : 1 pour "Rubik" et 97 : 1 pour "Taxi" n'ont jamais été préférées aux séquences reconstruites avec MPEG I. La raison de ceci est que le pourcentage des images de mise à jour pour ces deux séquences est de 6.7% et 8.3% respectivement comparé à 10% et 14% pour la plus grande compression des séquences "Nasa" et "Tunnel". Cette faible fréquence de transmission des images de mise à jour cause une accumulation d'erreur que les sujets ont jugé génante. Un exemple de ces accumulations d'erreur est montré aux figures 5.3(b) et 5.5(b) pour les séquences "Rubik" et "Taxi" respectivement.

Pour la séquence "Taxi", le premier et le deuxième sujets préféraient MPEG I car notre système générait des artefacts aux occlusions. En effet, quand un objet se déplace devant un fond statique, il n'est pas possible de reconstruire le fond que cet objet occupait et il faut attendre la prochaine image de mise à jour pour obtenir

les bonnes intensités lumineuses aux positions où l'objet est passé. Ceci fait, pour la séquence "Taxi", que la voiture noire au bas gauche de l'image laisse des traces noires derrière elle comme nous pouvons le voir à la figure 5.5(c). Le troisième sujet a confirmé avoir perçu ces traces mais a jugé qu'elles étaient moins gênante que l'effet de bloc sur les images reconstruites avec MPEG I (figure 5.5(a)). Les évaluations faites par ce sujet pour la séquence "Taxi" ont été, par conséquent, substentiellement différentes des deux autres sujets.

Au tableau 5.12, nous avons les compressions (T_{Seuil}) à partir desquelles les sujets commencent à préférer les images reconstruites avec notre système à celles de MPEG I et le gain de compression correspondant. Ces résultats montrent que la supériorité de notre système à MPEG I est plus accentuée pour les séquences "Taxi" et "Tunnel". Ceci confirme les résultats objectifs obtenus en se basant sur le RSBC (figure 5.1) et que nous avons discuté au préalable.

Tableau 5.12 : Compressions à partir desquelles le RSBC obtenu avec notre système commence à être supérieur à celui obtenu avec MPEG I

Séquence	T_{Seuil}	Gain de compression
"Rubik"	79 : 1	20%
"Taxi"	93 : 1	45%
"Nasa"	67 : 1	24%
"Tunnel"	69 : 1	42%

Pour évaluer la qualité des séquences reconstruites par rapport aux séquences originales, nous avons utilisé le critère d'évaluation de la dégradation en 7 points. Chacune des trois séquences reconstruites avec notre système (avec trois taux de

Tableau 5.13 : Résultats subjectifs

Compression	Premier sujet	Deuxième sujet	Troisième sujet	Moyenne
Séquence "Rubik"				
92 : 1	0/4	0/4	0/4	0/12
79 : 1	4/4	1/4	2/4	7/12
66 : 1	4/4	4/4	4/4	12/12
Séquence "Taxi"				
97 : 1	0/4	0/4	0/4	0/12
93 : 1	2/4	1/4	3/4	6/12
88 : 1	3/4	2/4	4/4	9/12
Séquence "Nasa"				
73 : 1	4/4	3/4	3/4	10/12
67 : 1	4/4	4/4	4/4	12/12
58 : 1	4/4	4/4	4/4	12/12
Séquence "Tunnel"				
75 : 1	4/4	4/4	4/4	12/12
69 : 1	4/4	4/4	4/4	12/12
63 : 1	4/4	4/4	4/4	12/12

compressions différents) ainsi que la séquence originale ont été présentées deux fois à chaque observateur dans un ordre aléatoire. Les résultats obtenus sont illustrés par le tableau 5.14.

Encore une fois, l'évaluation de la séquence "Taxi" a été sévère surtout par les deux premiers sujets. Ceci est causé par les traces noires laissées derrière la voiture comme expliqué auparavant.

La faible fréquence de transmission des images de mise à jour pour les séquences "Rubik" compressé à un taux de 92 : 1 et "Taxi" compressé à un taux de 97 : 1 fait que ces deux séquences ont été évaluées comme étant extrèmement dérangeante.

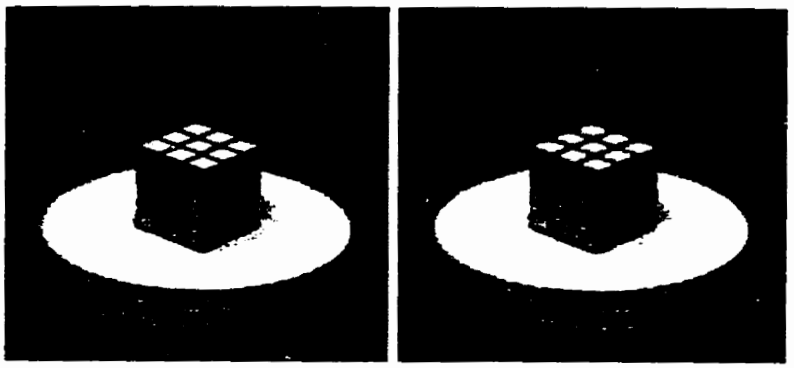
Le tableau 5.14 nous montre que les images reconstruites pour la séquence "Tun-

nel" avaient toujours des erreurs non perceptibles. Ces images d'excellente qualité sont montrées à la figure 5.9. Ceci nous indique qu'il aurait été possible de faire plus de compression sur cette séquence tout en obtenant une qualité acceptable des images reconstruites au récepteur.

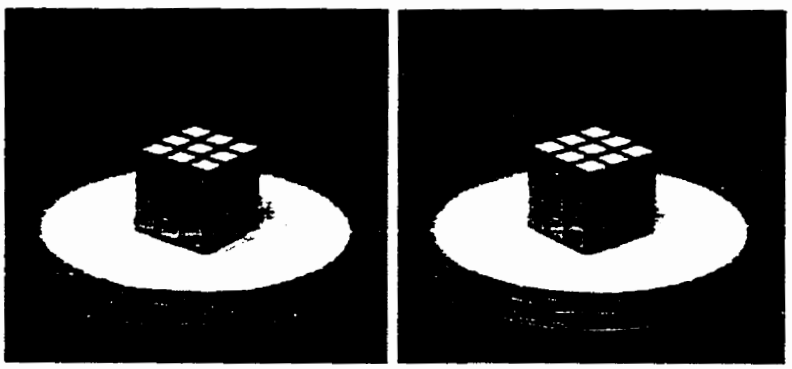
Il est à noter que les résultats obtenus sont subjectifs et diffèrent d'un sujet à l'autre. Dans notre cas, le troisième sujet a souvent été moins sévère que les deux autres.

Tableau 5.14 : Résultats comparatifs avec l'original

Séquence	Premier sujet	Deuxième sujet	Troisième sujet	Moyenne
Séquence "Rubik"				
Originale	1	1	2	1.33
92 : 1	7	6	4	5.67
79 : 1	4.5	5	2.5	4
69 : 1	1.5	1	1.5	1.33
Séquence "Taxi"				
Originale	1	1	1	1
97 : 1	7	7	7	7
93 : 1	6	5.5	4.5	5.33
88 : 1	4	3.5	3	3.5
Séquence "Nasa"				
Originale	1	1	1	1
73 : 1	2.5	2.5	3	2.67
67 : 1	2.5	2	1.5	2
58 : 1	2	1.5	1	1.5
Séquence "Tunnel"				
Originale	1	1	1	1
75 : 1	1	1	1	1
69 : 1	1	1	1	1
63 : 1	1	1	1	1

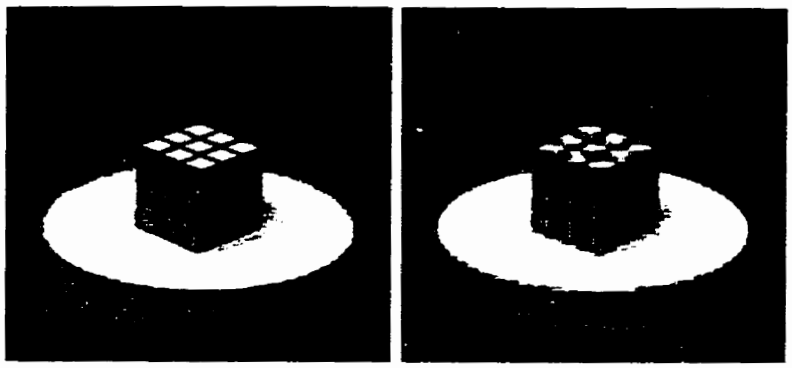


(a) MPEG I (66 : 1) (b) Notre système (92 : 1)

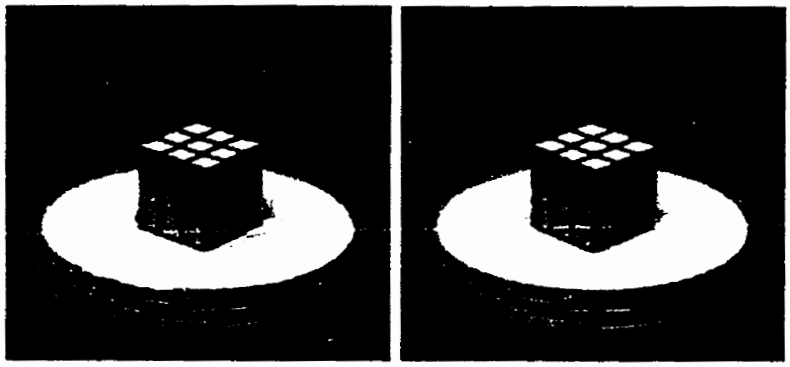


(c) Notre système (79 : 1) (d) Notre système (69 : 1)

Figure 5.2 : L'image reconstruite ayant une erreur moyenne pour la séquence "Rubik"

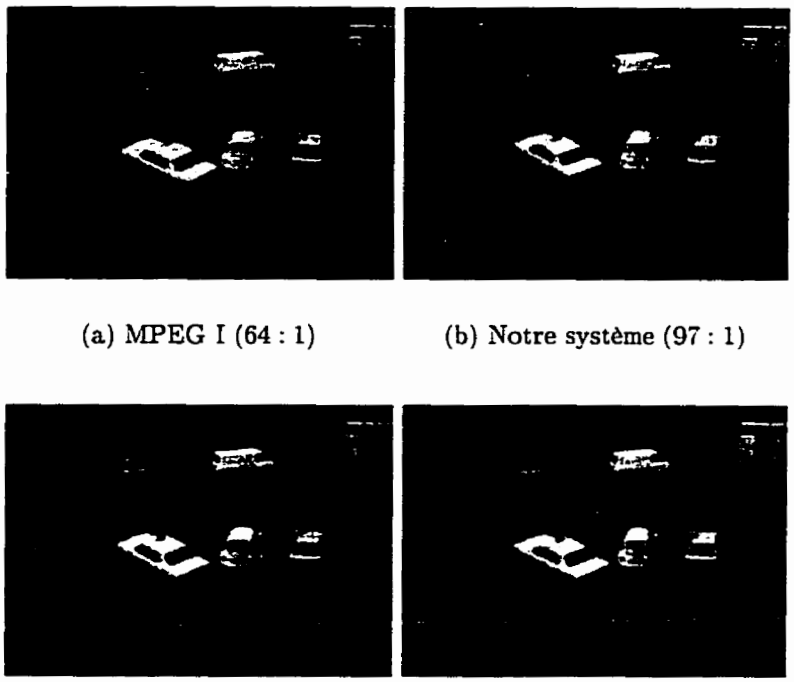


(a) MPEG I (66 : 1) (b) Notre système (92 : 1)



(c) Notre système (79 : 1) (d) Notre système (69 : 1)

Figure 5.3 : L'image reconstruite ayant une erreur maximale pour la séquence "Rubik"



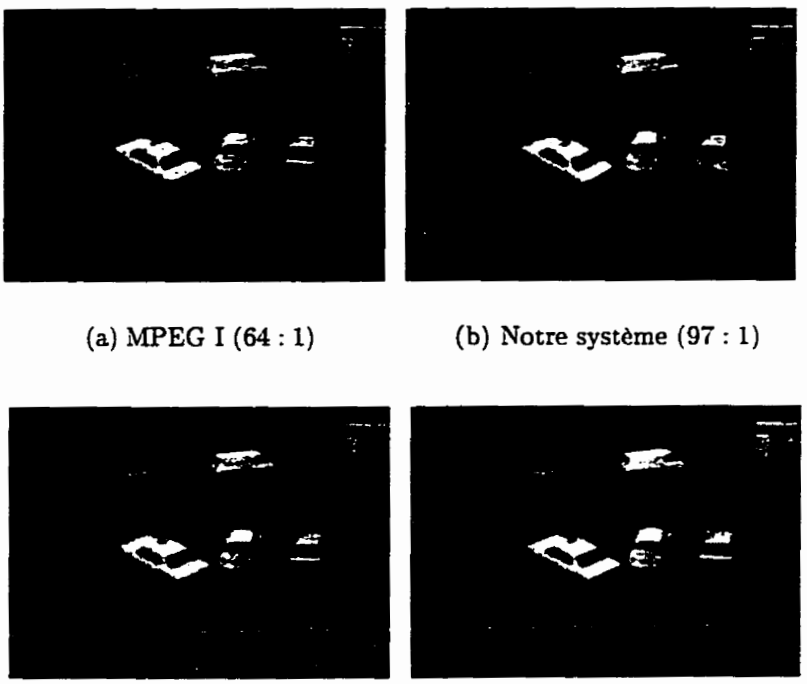
(a) MPEG I (64 : 1)

(b) Notre système (97 : 1)

(c) Notre système (93 : 1)

(d) Notre système (88 : 1)

Figure 5.4 : L'image reconstruite ayant une erreur moyenne pour la séquence "Taxi"



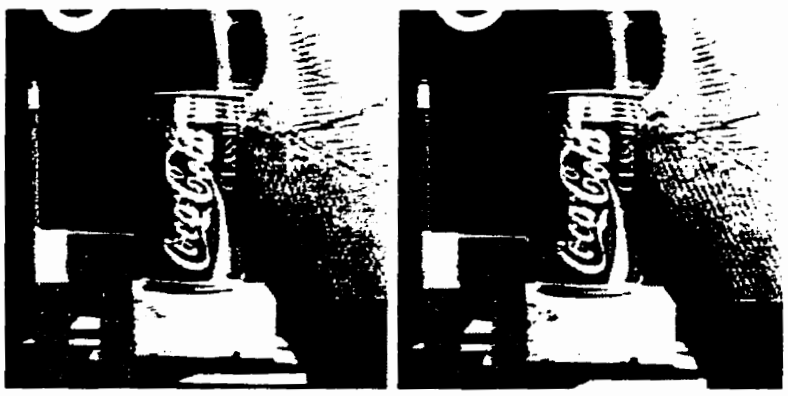
(a) MPEG I (64 : 1)

(b) Notre système (97 : 1)

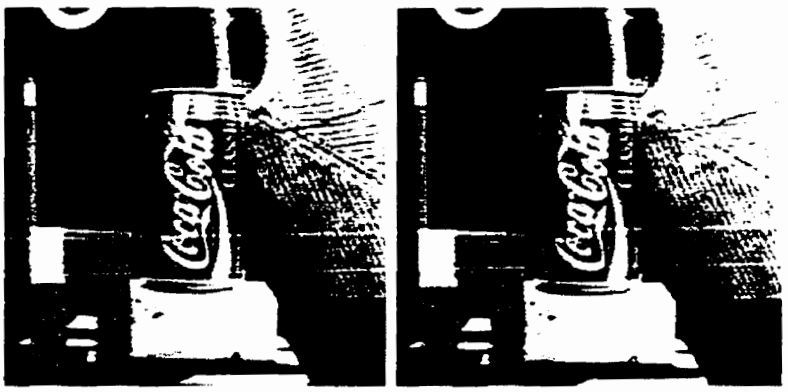
(c) Notre système (93 : 1)

(d) Notre système (88 : 1)

Figure 5.5 : L'image reconstruite ayant une erreur maximale pour la séquence "Taxi"

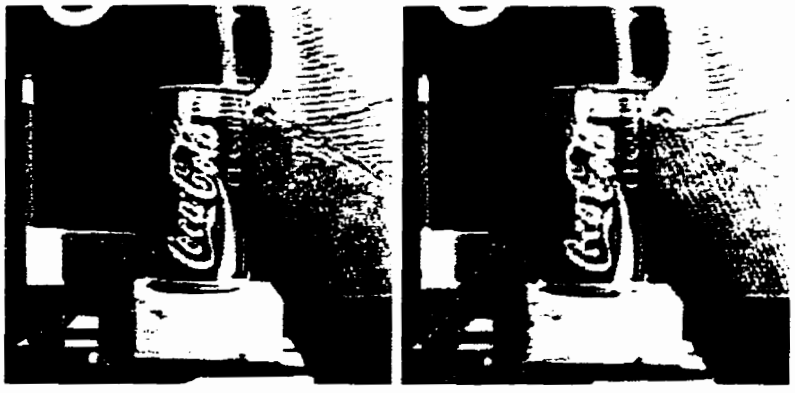


(a) MPEG I (59 : 1) (b) Notre système (73 : 1)

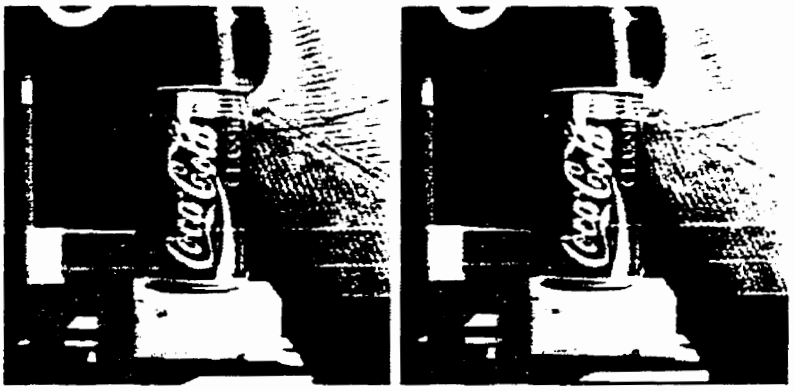


(c) Notre système (67 : 1) (d) Notre système (58 : 1)

Figure 5.6 : L'image reconstruite ayant une erreur moyenne pour la séquence "Nasa"

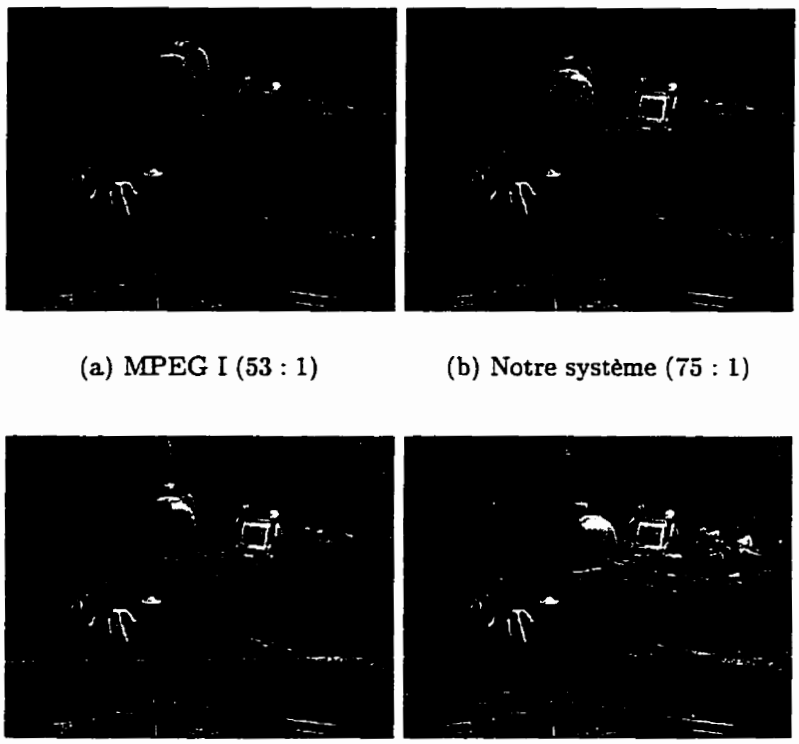


(a) MPEG I (59 : 1) (b) Notre système (73 : 1)



(c) Notre système (67 : 1) (d) Notre système (58 : 1)

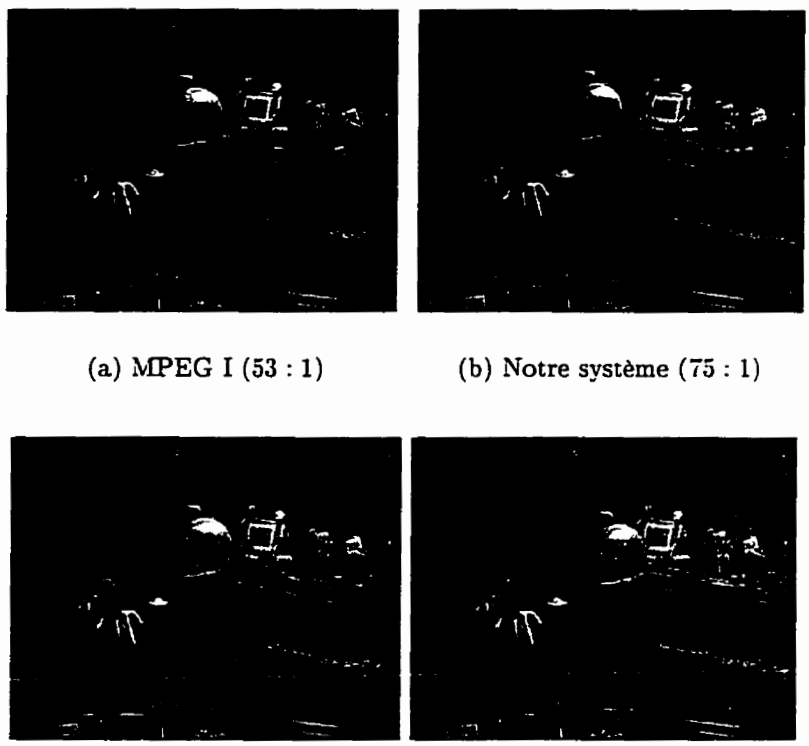
Figure 5.7 : L'image reconstruite ayant une erreur maximale pour la séquence "Nasa"



(a) MPEG I (53 : 1) (b) Notre système (75 : 1)

(c) Notre système (69 : 1) (d) Notre système (63 : 1)

Figure 5.8 : L'image reconstruite ayant une erreur moyenne pour la séquence "Tunnel"



(a) MPEG I (53 : 1) (b) Notre système (75 : 1)

(c) Notre système (69 : 1) (d) Notre système (63 : 1)

Figure 5.9 : L'image reconstruite ayant une erreur maximale pour la séquence "Tunnel"

Conclusion

Dans ce mémoire, nous avons développé un système de codage par compensation de mouvement pour représenter des séquences d'images monochromes à bas débit. Le système se base sur la modélisation du mouvement dans un domaine transformé avec des bases de transformation adaptées dans le temps par l'apprentissage Hebbien. Cette modélisation, présentée au chapitre 3, a été traitée en profondeur. Elle a été testée de manière satisfaisante sur des champs de mouvement simulés et réels en la comparant à la représentation du mouvement dans le domaine transformé par la TCD. Il ressort de ces tests que les bases adaptées permettent de modéliser le mouvement avec beaucoup plus de précision que les bases de la TCD pour le même nombre de paramètres utilisés dans le modèle.

Dans le chapitre 4, nous avons présenté le système de compensation de mouvement utilisant la modélisation par transformations adaptatives. Les flux reconstruits à partir des paramètres transmis du modèle sont utilisés pour reconstruire les images par interpolation des déplacements. Pour éviter l'accumulation des erreurs, des images de mise à jour sont codées dans le domaine spatial et transmises à chaque fois que

l'erreur dépasse un seuil.

Des tests numériques et psychophysiques, présentés au chapitre 5, ont démontrés que le système développé nous permet d'effectuer plus de compression que le standard MPEG pour une même qualité des images.

Ce mémoire présente trois innovations par rapport à ce qui a été fait précédemment dans la littérature scientifique:

- l'utilisation du mouvement dense pour faire la compensation du mouvement a permis d'obtenir des taux de compression suffisament élevés pour transmettre les séquences vidéo sur des canaux à bas débit;
- l'analyse spatiale des images a permis de ne pas avoir à coder les régions statiques;
- la quantité d'information utilisée pour représenter le flux optique aux blocs dynamiques est identique au déplacement par bloc (2 paramètres par bloc).

Le système de compression vidéo développé dans ce mémoire peut être étendu aux images en couleur. En effet, il suffit d'utiliser le mouvement reconstruit pour interpoler les trois composantes de chrominance au lieu d'effectuer cette interpolation uniquement sur la luminance. Il est évident que le taux de compression dans ce cas sera encore plus élevé puisque la quantité d'information à transmettre reste la même (information de mouvement) pour les images en couleur par rapport aux images monochromes.

Au lieu de retenir deux types d'images (images codées par compensation du mouvement et images de mise à jour), il aurait été possible d'utiliser deux types de blocs. En d'autres mots, on aurait pu coder tous les blocs sur lesquels la compensation de mouvement échoue dans le domaine spatial. Ceci nous aurait évité de transmettre des images complètes de mise à jour quand quelques blocs seulement sont erronés.

Comme nous l'avons exposé au chapitre 4, la principale source d'erreur est l'interpolation dans la reconstruction d'images. Il serait intéressant d'utiliser la position transmise des blocs dynamiques pour inhiber l'interpolation aux blocs statiques. Ceci est illustré à la figure 5.10. Au lieu d'interpoler l'intensité du sous-pixel (x, y) dans tout son voisinage, son intensité serait interpolée uniquement aux pixels hachurés appartenant à la région dynamique. D'autre part, Lin et Barron (Lin et Barron 1994) ont discuté du fait que la reconstruction directe est meilleure que la reconstruction inverse aux endroits où un objet bouge en face d'un autre et que la reconstruction inverse est meilleure aux endroits où un objet bouge derrière un autre. Il serait donc intéressant de détecter la position et le type d'occlusions et d'utiliser un système hybride qui choisirait entre la reconstruction directe et inverse selon le type d'occlusion.

Quoi qu'il en soit, la théorie présentée dans ce mémoire ouvre la voie à une toute nouvelle méthode de compensation de mouvement dense qui modélise ce mouvement dans un domaine transformé adapté dans le temps.

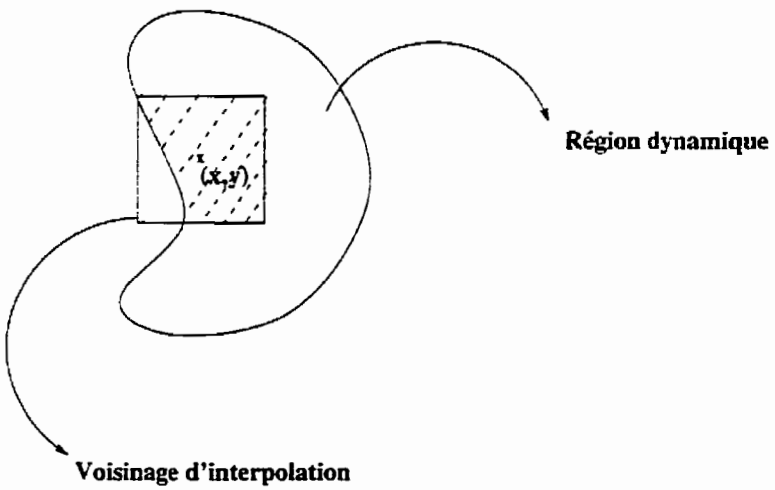


Figure 5.10 : Adaptation de l'interpolation aux positions des blocs dynamiques

Bibliographie

ALTUNBASAK, Y., TEKALP, A. M. et BOZDAGI, G. (1995), Two-dimensional object-based coding using a content-based mesh and affine motion parameterization, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 2, pp. 394–397.

ANANDAN, P. (1987), Measuring Visual Motion from Image Sequences, Thèse de doctorat, University of Massachusetts.

ANANDAN, P. (1989), A computational framework and an algorithm for the measurement of visual motion, *International Journal of Computer Vision* pp. 283–310.

BAAZIZ, N. et LABIT, C. (1994), Multiconstraint wiener-based motion compensation using wavelet pyramids, *IEEE Transactions on Image Processing* pp. 688–692.

BANHAM, M. R., BRAILEAN, J. C., CHAN, C. L. et KATSAGGELOS, A. K. (1994), Low bit rate video coding using robust motion vector regeneration in the decoder, *IEEE Transactions on Image Processing* **3**(5), 652–665.

BARTOLINI, F., CAPPELLINI, V., MECOCCI, A. et VAGHEGGI, R. (1994), A segmentation-based motion-compensated scheme for low-rate video coding, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 2, pp. 457–461.

BENOIT, S. M. et FERRIE, F. P. (1996), Monocular optical flow for real-time vision systems, dans *Proceedings of the International Conference on Pattern Recognition*, Vienna, Austria.

BERGMANN, H. C. (1982), Displacement estimation based on the correlation of image segments, dans *IEEE Proceedings of the International Conference on Electronic Image Processing*, pp. 215–219.

BERGMANN, H. C. (1984), Ein Schnell Konvergierendes Displacement-Schätzverfahren Für die Interpolation von Fernsehbild-Sequenzen, Thèse de doctorat, University of Hannover.

BHASKARAU, V. et KOUSTANTINIDES, K. (1995), *Image and Video Compression Standards*, Vol. 2, Kluwer Academic Pub.

BIEMOND, J., LOOIJENGA, L., BOEKEE, D. E. et PLOMPEN, R. H. J. M. (1987), A pel-recursive wiener-based displacement estimation algorithm, *Signal Processing* **13**, 399–412.

BONNAUD, L. et LABIT, C. (1994), Etude d'algorithmes de suivi temporel de segmentation basée mouvement pour la compression de séquences d'images, rapport technique 2253, INRIA.

BOUTHEMY, P. et FRANCOIS, E. (1993), Motion segmentation and qualitative dynamic scene analysis from an image sequence, *International Journal of Computer Vision* **10**(2), 157–182.

BURGER, W. et BHANU, B. (1992), *Qualitative Motion Understanding*, Kluwer academic publishers.

CAFFORIO, C. et ROCCA, F. (1983), The differential method for image motion estimation, *Image Sequence Processing and Dynamic Scene Analysis* pp. 104–124.

CAMUS, T. (1994), Real-Time Optical Flow, Thèse de doctorat, Brown University.

CARPENTIERI, B. et STORER, J. A. (1992), A split-merge parallel block-matching algorithm for video displacement estimation, dans *Data Compression Conference*, pp. 239–248.

CARPENTIERI, B. et STORER, J. A. (1994), Split-merge video displacement estimation, *Proceedings of the IEEE* **82**(6), 940–947.

CHAE, S. B., KIM, J. S. et PARK, R. H. (1993), Video coding by segmenting motion vectors and frame differences, *Optical Engineering* **32**(4), 870–876.

CHAN, M. H., YU, Y. B. et CONSTANTINIDES, A. G. (1990). Variable size block matching motion compensation with applications to video coding, dans *IEEE Proceedings. Part 1, Communications, Speech and Vision*, Vol. 137, pp. 205–212.

DANG, V. N., MANSOURI, A. R. et KONRAD, J. (1995). Motion estimation for region-based video coding, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 2, pp. 189–192.

DEMICHELI, E., SANDINI, G., TISTARELLI, M. et TORRE, V. (1988). Estimation of visual motion and 3d motion parameters from singular points, dans *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems*.

DUFAUX, F. et MOSCHENI, F. (1995). Motion estimation techniques for digital tv: A review and a new contribution. *Proceedings of the IEEE* **83**(6), 858–876.

DUFAUX, F., MOSCHENI, F. et LIPPMAN, A. (1995). Spatio-temporal segmentation based on motion and static segmentation, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 1, pp. 306–309.

FLEET, D. J. et JEPSON, A. D. (1990). Computation of component image velocity from local phase information. *International Journal Of Computer Vision* **5**(1), 77–104.

FULDSETH, A. et RAMSTAD, T. A. (1995), A new error criterion for block-based motion estimation, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 3, pp. 188–191.

GALL, D. L. (1991), Mpeg: A video compression standard for multimedia applications, *Communications of the ACM* **34**(4), 47–58.

GÍSLADÓTTIR, J. V. et ORCHARD, M. T. (1994), Motion-only video compression, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 3, pp. 730–734.

HADDADI, N. et KUO, C. C. J. (1994*a*), Motion compensated video compression and dense motion field coding, dans *Proceedings of the IEEE International Symposium on Circuits and Systems*, Vol. 3, pp. 213–216.

HADDADI, N. et KUO, C. C. J. (1994*b*), Multiple bit-rate video compression via progressive motion field coding, dans *IS&T/SPIE Symposium on Electronic Imaging: Science and Technology*.

HANG, H. M., SCHILLING, D. L. et PURI, A. (1987), Interframe coding with variable block-size motion compensation, dans *IEEE Global Telecommunications Conference*, pp. 65–69.

HEBB, D. (1949), *The Organisation of Behaviour*, New York : Wiley.

HORN, B. (1986), *Robot Vision*, The MIT Press.

HORN, B. K. P. et SCHUNCK, B. G. (1981), Determining optical flow. *Artificial Intelligence* pp. 185–204.

HUANG, J. et MERSEREAU, R. M. (1994), Multi-frame pel-recursive motion estimation for video image interpolation, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 2, pp. 267–271.

HUANG, S. Y., CHEN, J. R., WANG, J. S., HSIEH, K. R. et HSIEH, H. Y. (1994), Classified variable block size motion estimation algorithm for image sequence coding, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 3, pp. 736–740.

HUANG, Y. et ZHUANG, X. (1995), An adaptively refined block matching algorithm for motion compensated video coding, *IEEE Transactions on Circuits and Systems for Video Technology* **5**(1), 56–59.

ISHIGURO, T. et INUMA, K. (1982), Television bandwidth compression transmission by motion compensated interframe coding, *IEEE Communications Magazine* pp. 24–30.

ISO (1990), Coding of moving pictures and associated audio. *Committee Draft of Standard ISO11172: ISO/MPEG 90/176* .

JAIN, J. R. et JAIN, A. K. (1981), Displacement measurement and its application in interframe image coding, *IEEE Transactions on Communications* **29**(12), 1799–1808.

JOHNSON, L. W. et RIESS, R. D. (1982), *Numerical Analysis*, Addison-Wesley.

JOLLIFFE, I. T. (1986), *Principal Component Analysis*, Springer-Verlag.

KASS, M., WIKIN, A. et TERZOPOULOS, D. (1988), Snakes: Active contour models, *International Journal of Computer Vision* **1**, 321–331.

KRISHNAMURTHY, R., MOULIN, P. et WOODS, J. (1995), Optical flow techniques applied to video coding, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 1, pp. 570–573.

LEE, J. (1995), Optimal quadtree for variable block size motion estimation, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 3, pp. 480–483.

LEE, J. S. (1980), Digital image enhancement and noise filtering by use of local statistics, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2**(2), 165–168.

LEONARDI, R. et CHEN, H. (1994), Tree based motion compensated video coding, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 2, pp. 438–442.

LI, H., LUNDMARK, A. et FORCHHEIMER, R. (1994), Image sequence coding at very low bitrates: A review, *IEEE Transactions on Image Processing* **3**(5), 589–609.

LIN, T. et BARRON, J. L. (1994), Image recontruction error for optical flow, rapport technique, University of Western Ontario.

LUCAS, B. et KANADE, T. (1981), An iterative image registration technique with an application to stereo vision, dans *Proceedings DARPA Image Understanding Workshop*, pp. 121–130.

MATTHEWS, K. E. et NAMAZI, N. M. (1995), Simultaneous motion parameter estimation and image segmentation using the em algorithm, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 1, pp. 542–545.

MOCCAGATTA, I., MOSCHENI, F., SCHÜTZ, M. et DUFAUX, F. (1994), A motion field segmentation to improve moving edges reconstruction in video coding, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 3, pp. 751–755.

MOULIN, P. et LOUI, A. (1993), Application of a multiresolution optical-flow-based method for motion estimation to video coding, dans *IEEE International Symposium on Circuits and Systems*, Vol. 1, pp. 1–4.

MURAYAMA, J., MIYAUCHI, T. et SHIROTA, N. (1995), Image sequence coding using a contour-based method, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 1, pp. 546–549.

MUSMANN, H. G., PIRSH, P. et GRALLERT, H. J. (1985), Advances in picture coding, *Proceedings of the IEEE* **73**(4), 523–548.

NAGEL, H. H. (1983), Displacement vectors derived from second-order intensity variations in image sequences, *Computer Graphics, Image Processing* pp. 85–117.

NAGEL, H. H. (1987), On the estimation of optical flow: Relations between different approches and some new results, *Artificial Intelligence* pp. 299–324.

NAGEL, H. H. et ENKELMANN, W. (1986), An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences, *IEEE Transactions on Pattern Analysis and Machine Intelligence* pp. 565–593.

NAKAJIMA, Y., HORI, H. et KANOH, T. (1994), A pel adaptive reduction of coding artifacts for mpeg video signals, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 2, pp. 928–932.

NAKAYA, Y. et HARASHIMA, H. (1994), Motion compensation based on spatial transformations, *IEEE Transactions on Circuits and Systems for Video Technology* 4(3), 339–356.

NAMAZI, N. M., PENAFIEL, P. et FAN, C. M. (1994), Nonuniform image motion estimation using kalman filtering, *IEEE Transactions on Image Processing* **3**(5), 678–683.

NETRAVALI, A. N. et ROBBINS, J. D. (1979), Motion compensated television coding – part I, *Bell Systems Technology Journal* **58**, 631–670.

NIEWEGLOWSKI, J., CAMPBELL, T. G. et HAAVISTO, P. (1993), Novel video coding scheme based on temporal prediction using digital image warping, *IEEE Transactions on Consumer Electronics* **39**(3), 141–150.

OHTA, M. et NOGAKI, S. (1993), Hybrid picture coding with wavelet transform and overlapped motion-compensated interframe prediction coding, *IEEE Transaction on Signal Processing* **41**(12), 3416–3424.

OJA, E. (1982), A simplified neuron model as a principal component analyzer, *Journal of Mathematical Biology* **15**, 267–273.

OPPENHEIM, A. V. et LIM, J. S. (1981), The importance of phase in signals, *Proceedings of the IEEE* **69**(5), 529–541.

ORCHARD, M. T. (1993), Predictive motion-field segmentation for image sequence coding, *IEEE Transactions on Circuits and Systems for Video Technology* **3**(1), 54–70.

PAPADOPOULOS, C. A. et CLARKSON, T. G. (1992), Motion compensation using 2nd order geometric transformations, *Electronics Letters* **28**(25), 2314–2315.

PAPADOPOULOS, C. A. et CLARKSON, T. G. (1993), Use of 2nd order geometric transformations for motion compensation in interframe image data compression, dans *Proceedings of the IEEE Global Telecommunications Conference*, Vol. 2, pp. 1309–1313.

PARDÀS, M., SALEMBIER, P. et GONZALEZ, B. (1994), Motion and region overlapping estimation for segmentation-based video coding, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 2, pp. 428–432.

RAJALA, S., CIVANLAR, M. et LEE, W. (1988), Video data compression using three-dimensional segmentation based on human visual system properties, dans *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing*, pp. 1092–1095.

RAO, K. R. et YIP, Y. (1990), *Discrete Cosine Transform: Algorithms, Advantages, Applications*, New York: Academic.

SALARI, E. et LIN, S. (1995), Low-bit-rate segmentation-based image sequence coding, *Optical Engineering* **34**(3), 829–833.

SALEMBIER, P., GU, C., PARDÀS, M. et KUNT, M. (1994), Very low bit rate video coding using morphological segmentation and contour texture motion compensation, dans *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, Vol. 3, pp. 25–29.

SALEMBIER, P., TORRES, L., MEYER, F. et GU, C. (1995), Region-based video coding using mathematical morphology, *Proceedings of the IEEE* **83**(6), 843–857.

SANGER, T. D. (1989), Optimal unsupervised learning in a single-layer linear feedforward neural network, *Neural Networks* **2**, 459–473.

SINGH, A. (1992), *Optic Flow Computation: A Unified Perspective*. IEEE Computer Society Press.

SULLIVAN, G. J. et BAKER, R. L. (1991). Rate-distortion optimized motion compensation for video compression, dans *IEEE Global Telecommunications Conference*, Vol. 1, pp. 85–90.

TSAI, R. Y. et HUANG, T. S. (1981), Estimating three-dimensional motion parameters of a rigid planar patch, *IEEE Transactions on Acoustics Speech and Signal Processing* **29**(6).

WALKER, D. R. et RAO, K. R. (1984). Improved pel-recursive motion compensation. *IEEE Transactions on Communications* **32**(10), 1128–1134.

WANG, Y., HSIEH, X. M., HU, J. H. et LEE, O. (1995). Region segmentation based on active mesh representation of motion: Comparision of parallel and sequential approaches, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 2, pp. 185–188.

WANG, Y. et LEE, O. (1994), Active mesh – a feature seeking and tracking image sequence representation scheme, *IEEE Transactions on Image Processing* **3**(5), 610–624.

WENG, J., AHUJA, N. et HUANG, T. (1992), Matching two perspective views, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14**(8), 806–825.

WENG, J. J. (1993), Image matching using the windowed fourier phase, *International Journal of Computer Vision* **11**(3), 211–236.

WIEGAND, T., LIGHTSTONE, M., CAMPBELL, T. G. et MITRA, S. K. (1995), Efficient mode selection for block-based motion compensated video coding, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 2, pp. 559–562.

WU, L., BENOIS-PINEAU, J. et BARBA, D. (1995), Spatio-temporal segmentation of image sequences for object-oriented low bit-rate image coding, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 2, pp. 406–409.

YEH, J., VETTERLI, M. et KHANSARI, M. (1995), Motion compensation of motion vectors, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 1, pp. 574–577.

YOUNG, R. W. et KINGSBURY, N. G. (1993), Video compression using lapped transforms for motion estimation/compensation and coding, *optical Engineering* **32**(7), 1451–1463.

YSERENTANT, H. (1986), On the multilevel splitting of finite element spaces, *Numerische Mathematik* **49**, 379–412.

ZAFAR, S., ZHANG, Y. Q. et JABBARI, B. (1993), Multiscale video representation using multiresolution motion compensation and wavelet decomposition. *IEEE Journal on Selected Areas in Communications* **11**(1), 24–35.

ZHANG, K., BOBER, M. et KITTLER, J. (1995), Motion based image segmentation for video coding, dans *Proceedings of the IEEE International Conference on Image Processing*, Vol. 3, pp. 476–479.

ZHENG, H. et BLOSTEIN, S. D. (1995), Motion-based object segmentation and estimation using the MDL principle. *IEEE Transactions on Image Processing* **4**(9), 1223–1235.

ZSCHUNKE, W. (1977), Dpcm picture coding with adaptive prediction. *IEEE Transactions on Communications* **25**(11), 1295–1302.

Annexe A

Revue bibliographique:

Compléments

Cette annexe vient compléter le chapitre 1 en présentant le détail des méthodes existant dans la littérature.

A.1 Modélisation du mouvement

Avant d'expliquer les modèles de mouvement utilisés pour le codage des séquences d'images, il est important de comprendre la relation entre le mouvement de la scène et le champ de mouvement.

Nous allons considérer le champ de mouvement dans une région[1] de l'image qui correspond à un mouvement rigide dans la scène. Ce mouvement rigide peut être

[1] Une région peut correspondre à un bloc ou avoir une forme arbitraire

représenté par une translation $\vec{T} = (T_1, T_2, T_3)^T$ et une rotation $\vec{\Omega} = (\omega_1, \omega_2, \omega_3)^T$. Nous allons, aussi, supposer que la caméra nous fournit une projection idéale en perspective. Cette projection peut être décrite selon les équations:

$$x = f\frac{X}{Z} \tag{A.1}$$

$$y = f\frac{Y}{Z} \tag{A.2}$$

où f est la distance focale de la caméra.

Considérons un point $P_0 = (X, Y, Z)$ de la scène qui se déplace au point $P_1(X', Y', Z')$. La projection du premier point dans le plan image est $p_0(x, y)$ et du deuxième est $p_1(x', y')$. Burger et Bhanu (1992) ont développé les équations reliant les positions possibles de p_1 par rapport à p_0 sachant les paramètres de mouvement ($\vec{T}$ et $\vec{\Omega}$) et la profondeur du point (Z). Ils ont fait l'hypothèse que la rotation par rapport à l'axe de la caméra est nulle ($\omega_3 = 0$)[2]. Bien sûr, nous ne pouvons pas faire cette hypothèse, nous allons donc généraliser pour un ω_3 quelconque.

Le mouvement tri-dimensionnel entre P_0 et P_1 peut être décrit selon l'équation:

$$P_1 = R(\Omega) \times (T + P_0) \tag{A.3}$$

où $R(\Omega)$ est la matrice de rotation correspondant à un angle ω_1 par rapport à X suivi par une rotation de ω_2 par rapport à Y et enfin une rotation de ω_3 par rapport à Z[3].

[2]Leur étude se limitait au cas d'un véhicule autonome ce qui justifie cette hypothèse

[3]La rotation par rapport à X affecte l'orientation des axes Y et Z et la rotation par rapport à Y

Cette matrice peut être exprimée selon l'équation:

$$R(\Omega) = \begin{bmatrix} C_{\omega_2}C_{\omega_3} & -C\omega_2 S\omega_3 & S_{\omega_2} \\ S_{\omega_1}S_{\omega_2}C_{\omega_3} + C_{\omega_1}S_{\omega_3} & -S_{\omega_1}S_{\omega_2}S_{\omega_3} + C_{\omega_1}C_{\omega_3} & -S_{\omega_1}C_{\omega_2} \\ -C_{\omega_1}S_{\omega_2}C_{\omega_3} + S_{\omega_1}S_{\omega_3} & C_{\omega_1}S_{\omega_2}S_{\omega_3} + S_{\omega_1}C_{\omega_3} & C_{\omega_1}C_{\omega_2} \end{bmatrix} \quad (A.4)$$

où S_x et C_x représentent $\sin x$ et $\cos x$ respectivement.

En remplaçant les équations de projection en perspective A.1 et A.2 dans l'équation de mouvement A.3, nous obtenons:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \frac{f}{(-C_{\omega_1}S_{\omega_2}C_{\omega_3} + S_{\omega_1}S_{\omega_3})\,\dot{x} + (C_{\omega_1}S_{\omega_2}S_{\omega_3} + S_{\omega_1}C_{\omega_3})\,\dot{y} + C_{\omega_1}C_{\omega_2}\,\dot{f}} \begin{bmatrix} C_{\omega_2}C_{\omega_3}\,\dot{x} - C_{\omega_2}S_{\omega_3}\,\dot{y} + S_{\omega_2}\,\dot{f} \\ (S_{\omega_1}S_{\omega_2}C_{\omega_3} + C_{\omega_1}S_{\omega_3})\,\dot{x} + (-S_{\omega_1}S_{\omega_2}S_{\omega_3} + C_{\omega_1}C_{\omega_3})\,\dot{y} - S_{\omega_1}C_{\omega_2}\,\dot{f} \end{bmatrix} \quad (A.5)$$

$$\dot{x} = x + \frac{fT_1}{Z} \quad (A.6)$$

$$\dot{y} = y + \frac{fT_2}{Z} \quad (A.7)$$

$$\dot{f} = f + \frac{fT_3}{Z} \quad (A.8)$$

où $(\dot{x}, \dot{y})$ correspond à la position du point (x, y) dans le plan image suite à la translation fronto-parallèle. D'autre part, $\frac{f}{\dot{f}}(x, y)$ donne la position du point (x, y) suite à un mouvement purement divergent.

---

affecte l'orientation de l'axe Z

L'équation A.5 décrit le mouvement rigide d'un objet. Si nous segmentons les images en régions subissant des mouvement rigides, l'information de segmentation et les paramètres du modèle pour chacune des régions sont suffisant pour décrire le mouvement dense. Ce modèle s'appelle le modèle homographique, nous le décrivons dans la section A.1.1.

Le bruit présent sur les images cause que les paramètres de ce modèle sont difficiles à estimer en pratique. Par conséquent, Plusieurs modèles ont été proposés pour remplacer le modèle homographique. Quand le nombre de paramètres du modèle augmente, il devient plus représentatif du champ de mouvement mais il devient aussi plus compliqué à estimer et plus coûteux à décrire. Le nombre minimal de vecteurs nécessaires pour estimer les paramètres du modèle est la moitié du nombre de ces paramètres (chaque vecteur fournit deux équations, une pour chaque composante). En pratique, nous utilisons plus de vecteurs pour diminuer la sensibilité de ce calcul au bruit. Dans ce qui suit, nous allons décrire et comparer plusieurs modèles qui sont souvent utilisés dans la littérature.

A.1.1 Modèle homographique à 8 paramètres

Ce modèle se base directement sur la projection du mouvement tri-dimensionnel sur le plan image (équation A.5). Il est décrit selon les équations suivantes (Tsai et

Huang 1981):

$$\begin{aligned} x' &= \frac{a_1 x + a_2 y + a_3}{a_7 x + a_8 y + 1} \\ y' &= \frac{a_4 x + a_5 y + a_6}{a_7 x + a_8 y + 1} \end{aligned} \tag{A.9}$$

Bien que ce modèle décrit exactement le mouvement de la scène, il est rarement utilisé en codage (Nakaya et Harashima 1994). La raison principale de ceci est la complexité des calculs de ses paramètres. Dans tous les modèles qui suivent, le mouvement va être représenté par des polynômes. En effet, l'estimation des paramètres des polynômes est une tâche qui ne requiert pas beaucoup de calcul.

A.1.2 Modèle polynômial à 12 paramètres

Ce modèle décrit les mouvements complexes de translation, rotation, mise à l'échelle, cisaillement et déformation. Il est décrit selon les équations suivantes (Papadopoulos et Clarkson 1992, Papadopoulos et Clarkson 1993):

$$\begin{aligned} x' &= a_1 x^2 + a_2 x + a_3 xy + a_4 y^2 + a_5 y + a_6 \\ y' &= b_1 x^2 + b_2 x + b_3 xy + b_4 y^2 + b_5 y + b_6 \end{aligned} \tag{A.10}$$

Ce modèle est rarement utilisé à cause des deux raisons suivantes (Bonnaud et Labit 1994):

1. les termes quadratiques sont difficiles à estimer;

2. le nombre élevé des paramètres de ce modèle nous oblige de nous baser sur des plus grandes régions pour estimer avec exactitude ses paramètres et diminuer l'entête à transmettre pour l'information de mouvement ce qui rend l'hypothèse de rigidité plus difficile à vérifier.

A.1.3 Modèle bilinéaire à 8 paramètres

En éliminant les termes quadratiques du modèle polynômial à 12 paramètres, nous trouvons le modèle biblinéaire décrit selon les équations suivantes (Wang et al. 1995):

$$\begin{aligned} x' &= a_1 xy + a_2 x + a_3 y + a_4 \\ y' &= b_1 xy + b_2 x + b_3 y + b_4 \end{aligned} \tag{A.11}$$

Ce modèle est souvent utiliser pour représenter le mouvement de quadrilatères.

A.1.4 Modèle affine à 6 paramètres

Ce modèle décrit les mouvement de . Il est décrit selon l'équation suivante (Pardàs, Salembier et Gonzalez 1994, Zheng et Blostein 1995):

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x \cos\theta_x & -S_y \sin\theta_y \\ S_x \sin\theta_x & S_y \cos\theta_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \tag{A.12}$$

où $\vec{v} = (v_1, v_2)$ exprime une translation, $\vec{S} = (S_x, S_y)$ une homothétie et $\vec{\theta} = (\theta_x, \theta_y)$ une rotation.

Ce modèle est souvent utiliser pour représenter le mouvement de triangles.

A.1.5 Modèle affine à 4 paramètres

En supposant que la rotation et l'homothétie dans le modèle à six paramètres sont faibles ($\theta_x \approx 0$, $\theta_y \approx 0$ $S_x \approx 1$ et $S_y \approx 1$), nous pouvons faire les approximations suivantes (Bonnaud et Labit 1994):

$$S_x \cos\theta_x = S \tag{A.13}$$

$$S_y \cos\theta_y = S \tag{A.14}$$

$$S_x \sin\theta_x = \theta \tag{A.15}$$

$$S_y \sin\theta_y = \theta \tag{A.16}$$

et donc le modèle à 6 paramètres devient le suivant à 4 paramètres:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S & -\theta \\ \theta & S \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \tag{A.17}$$

A.1.6 Modèle translationnel à 2 paramètres

Ce modèle suppose que la région provient d'un objet de la scène qui subit une translation fronto-parallèle. Il est décrit selon l'équation suivante (Salembier. Torres. Meyer et Gu 1995):

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \tag{A.18}$$

Il est souvent utilisé dans les techniques de compensation de mouvement par bloc (section 1.3.1) et par régions polygonales (section 1.3.2.1).

A.1.7 Modèle nul (0 paramètres)

Ce modèle suppose que la région n'est pas en mouvement. Il est utilisé par exemple pour un fond immobile (Wiegand et al. 1995). Il faut bien sûr détecter ces zones (endroits où I_{t+1} est identique à I_t). Il est décrit par l'équation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{A.19}$$

A.2 Compensation de mouvement par bloc

A.2.1 Codage adaptatif par classification des blocs

Wiegand et al. (1995) proposent un algorithme de codage qui se base sur le même principe que MPEG mais qui permet de choisir le type des blocs de façon adaptative. Ainsi, deux blocs différents de la même image peuvent être attribués à deux types différents d'images. De plus, deux nouveaux types de blocs sont rajoutés:

1. **Blocs de type** $P4$ qui sont codés de la même manière que les blocs de type P mais avec le mouvement représenté par quatre vecteurs reliant ces blocs avec des sous-blocs de l'image précédente. Ceci revient à utiliser des blocs quatre fois plus petits et donc de relaxer la contrainte de mouvement translationnel à une région plus petite.

2. **Blocs de type** U qui sont des blocs de type P ayant un mouvement nul (section A.1.7). Ils sont alors codés de manière prédictive par rapport aux blocs à la même position de l'image précédente. Ceci a l'avantage de nous éviter de transmettre l'information de mouvement pour les blocs statiques.

Le choix d'un type de bloc en particulier se fait pour minimiser le rapport taux/distorsion exprimé avec la fonction $\min_c[D(c) + \lambda R(c)]$ où $D(c)$ est la distorsion causée par le codage du bloc selon le type c, $R(c)$ est le nombre de bits requis pour le codage du bloc selon ce type et λ est un multiplicateur de Lagrange. Il est à noter que pour les blocs de type P et $P4$, $R(c)$ dépend des modes utilisés pour coder les blocs adjacents

car les vecteurs de mouvement sont codés de façon différentielle spatiallement.

Huang et al. (1994) utilisent aussi ce principe de classification. Trois types de blocs sont retenus:

1. **Mouvement de fond** ces blocs sont proches de ceux à la même position dans l'image précédente et sont codés de la même manière que les blocs de type U qu'on vient de voir.

2. **Mouvement d'ombrage** ces blocs sont des blocs qui n'appartiennent pas au fond et qui ont une variance spatiale petite. Ils sont subdivisés en sous-blocs de 2×2 et on transmet un vecteur de déplacement par sous-bloc.

3. **Mouvement d'arêtes** on transmet le mouvement dense pour ces blocs. Les vecteurs de déplacement sont codés par un codage entropique $DPCM$. Ce codage n'exploite pas assez les redondances dans ce champ de vecteur ce qui se traduit en une faible compression.

La figure A.1 présente la structure du codeur.

A.2.2 Codage par blocs de taille variable

Zafar et al. (1993) proposent une méthode à multirésolution. Les images sont alors divisées en sous-bandes en utilisant la transformation en ondelettes. Ceci a pour effet que les blocs des bandes de hautes fréquences correspondent à des plus grandes régions dans les images originales. Comme nous pouvons le constater à l'exemple de

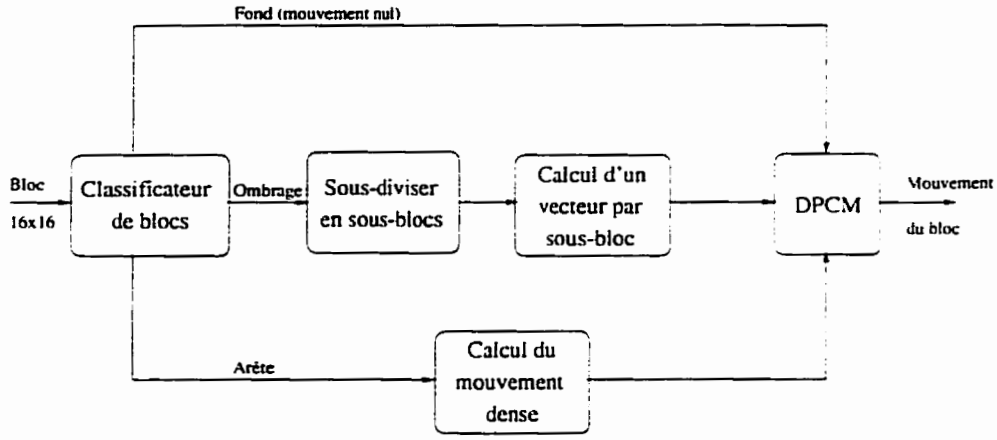


Figure A.1 : Le codeur-classifieur de Huang et *al.*

la figure A.2, les vecteurs de déplacement à un niveau sont estimés en raffinant les vecteurs projettés du niveau supérieur (de fréquence inférieure). Ils affirment qu'il vaut mieux appliquer la compensation de mouvement à chaque bande des images qu'appliquer une compensation de mouvement et de coder l'erreur de reconstruction en sous-bandes comme dans (Ohta et Nogaki 1993).

Hang et al. (1987) se basent, plutôt, sur un algorithme "grossier à fin" en commençant avec un bloc constituant l'image au complet. A chaque itération, on calcule les vecteurs de déplacement pour les blocs, puis si l'erreur de reconstruction (EQM) pour un bloc est trop grande, on le subdivise en plusieurs sous-blocs et on recommence le même type de procédure sur ces derniers. En basant la décision de subdiviser les blocs sur l'erreur de reconstruction, on ne tient pas compte de l'ajout de bits à transmettre pour la segmentation. On risque alors de descendre toujours à un même niveau de sous blocs, ce qui serait équivalent à prendre une taille fixe petite. Dufaux

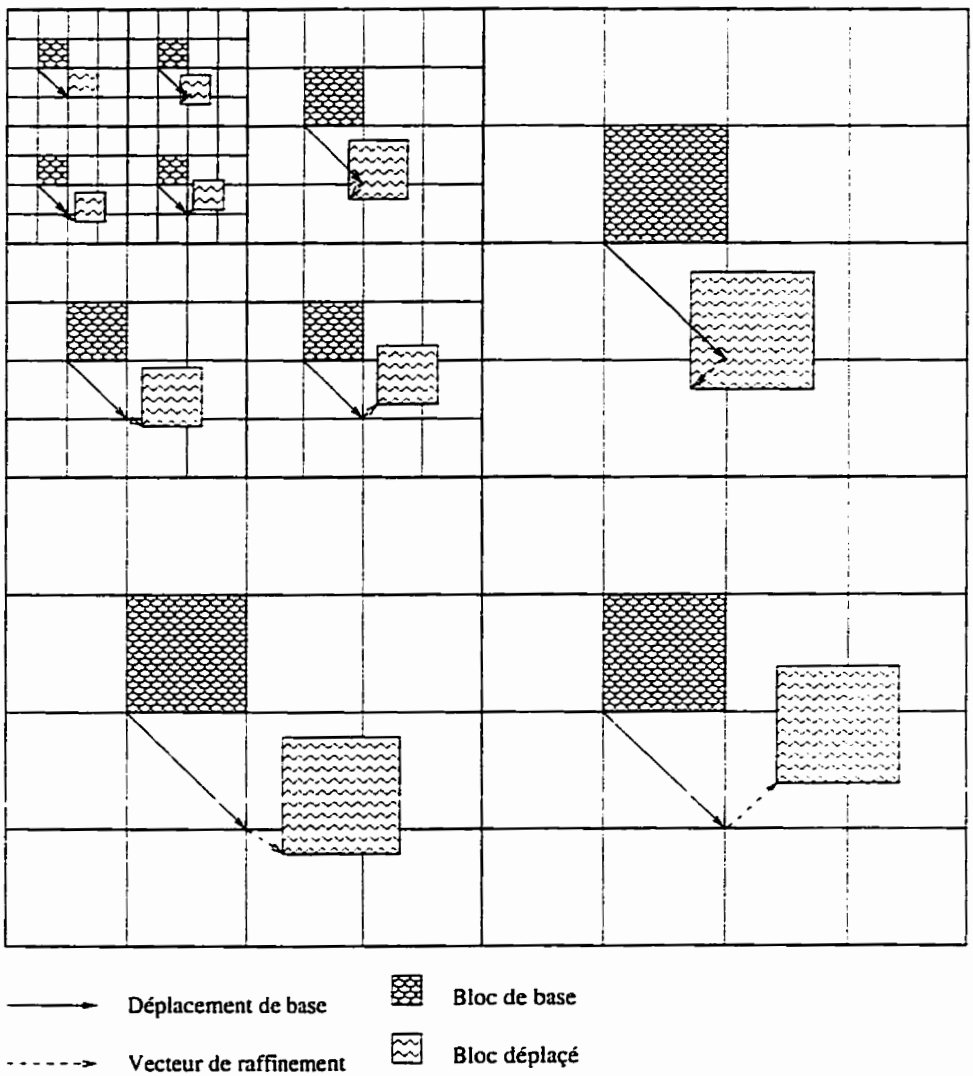


Figure A.2 : Estimation du mouvement en multirésolution

et Moscheni (1995) proposent de se baser plutôt sur la minimisation de l'entropie de l'information à transmettre. On divise alors un bloc de $N \times N$ en quatre si la diminution de l'entropie de l'erreur de reconstruction apportée par cette meilleure description du mouvement justifie l'augmentation de l'entropie de l'information de mouvement causée par le besoin de transmettre quatre vecteurs de mouvement au lieu d'un seul. Ceci peut être décrit selon le critère suivant:

$$N^2(H_{E_r\text{sans division}} - H_{E_r\text{division}}) > 4H_{\vec{v}\text{division}} - H_{\vec{v}\text{sans division}} \Rightarrow \text{subdiviser} \tag{A.20}$$

où H_x est l'entropie de x et E_r est l'erreur de reconstruction.

Par ailleurs, Sullivan et Baker (1991) voient le problème comme celui d'une allocation de bits. Par opposé à Dufaux et Moscheni (1995), ils utilisent une méthode "fin à grossier" et décident de fusionner quatre blocs (B_1, B_2, B_3 et B_4) en un seul (B) si ceci diminue la fonction suivante:

$$D(B) + \lambda R(B) < \sum_{i=1}^{4}[D(B_i) + \lambda R(B_i)] \Rightarrow \text{fusionner} \tag{A.21}$$

où $D(B)$ est une mesure de distorsion sur la reconstruction du bloc, $R(B)$ est le nombre de bits requis pour le codage de ce bloc et λ est un multiplicateur de Lagrange qui détermine l'importance relative du taux de compression et de la qualité des images désirée. Ce multiplicateur peut être choisi en fonction de l'application en cours.

A.2.3 Compensation par bloc basée sur une meilleure description du mouvement

Moccagatta et al. (1994) et Orchard (1993) se proposent d'améliorer les techniquesd'appariement de blocs, en permettant de séparer un bloc en deux régions et d'assigner pour chacune un vecteur de déplacement choisi à partir d'un voisinage du bloc en question. Ils supposent qu'il n'y a pas plus de deux régions dans un bloc. La différence entre (Moccagatta et al. 1994) et (Orchard 1993) provient de la manière de faire la segmentation des blocs. Dans (Moccagatta et al. 1994), on se base sur une table de recherche contenant plusieurs segmentations possibles obtenues dans le domaine spatial pour choisir celle qui minimise l'erreur de reconstruction. Tandis que, dans (Orchard 1993), un modèle markovien est utilisé pour représenter la segmentation[4]. Cette dernière sera estimée pour minimiser l'erreur de reconstruction tout en gardant une segmentation continue (deux régions séparées).

Fuldseth et Ramstad (1995) remarquent que le critère EQM utilisé dans les techniques d'appariement de blocs pour calculer les vecteurs de déplacement ne tient pas compte du codage spatial effectué sur les erreurs de reconstruction. En effet, le choix d'un vecteur qui minimise l'EQM peut amener à des erreurs de reconstruction moins corrélées spatiallement et donc plus difficile à coder, diminuant ainsi la qualité obtenue pour un taux de compression fixe. Dans cette optique, ils proposent d'utiliser le produit des variances des coefficients de la transformation TCD des erreurs de re-

[4]la segmentation d'un pixel dépend de celle de ses voisins

construction comme critère à minimiser pour le calcul des vecteurs de déplacement. Ils calculent ainsi la transformation TCD de l'erreur de reconstruction correspondant à chacun des vecteurs de déplacement dans l'espace de recherche et choisissent celui qui donne la plus petite valeur pour le produit des variances des coefficients de transformation. Pour pouvoir calculer la variance des coefficients. ils utilisent des plus petits blocs pour le codage de l'erreur de reconstruction que pour la compensation du mouvement pour obtenir ainsi plusieurs valeurs de l'erreur de reconstruction pour chaque vecteur de déplacement. Ceci a pour conséquence d'améliorer le rapport taux/distorsion du codeur.

A.2.4 Effet de bloc

Plusieurs méthodes ont été proposées pour réduire la visibilité de l'effet de blocs. Le concept de blocs superposés a été proposé (Ohta et Nogaki 1993. Young et Kingsbury 1993). Ces blocs sont pondérés par une fonction de fenêtrage qui accorde plus d'importance au centre du bloc qu'à sa bordure(figure A.3). Cette pondération est nécessaire pour accorder la même importance aux pixels appartenant à plusieurs blocs qu'aux pixels qui n'appartiennet qu'à un seul bloc. Young et Kingsbury (1993) proposent. pour des blocs de $2N \times 2N$, la fonction de fenêtrage suivante:

$$w(x,y) = \cos^2(\frac{\pi x}{2N}) \cos^2(\frac{\pi y}{2N}) \tag{A.22}$$

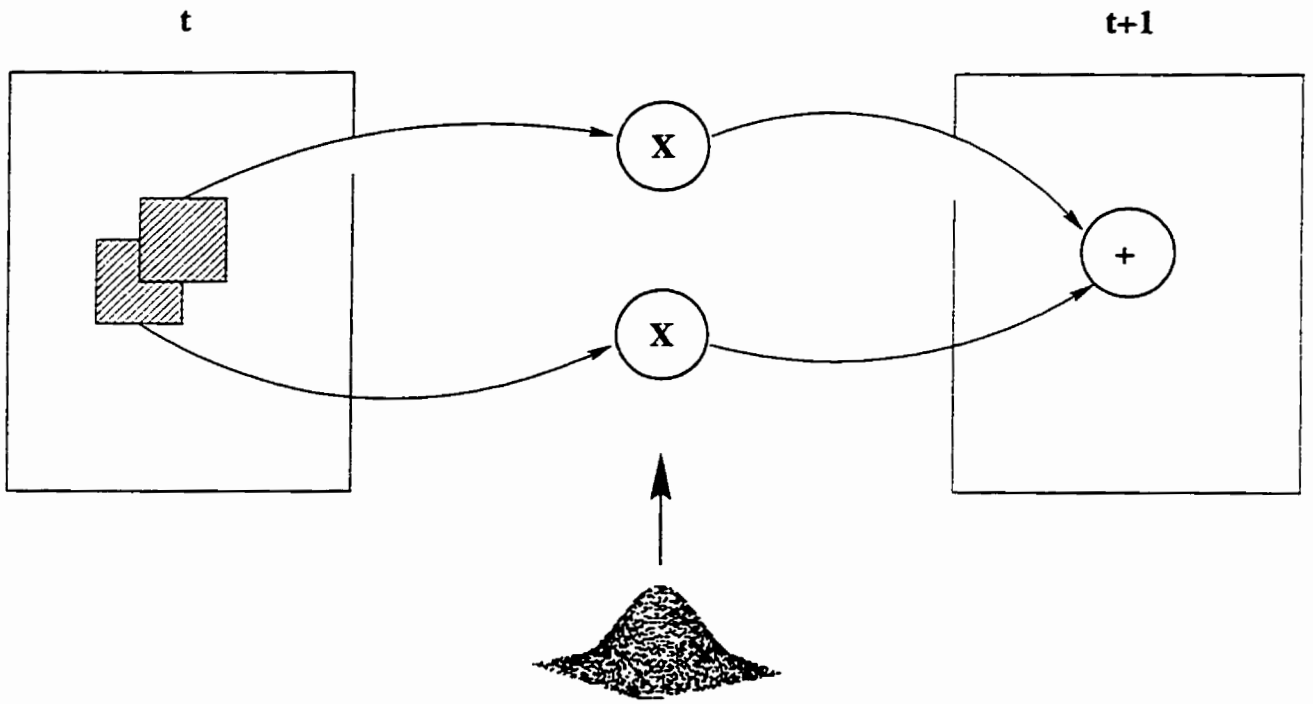


Figure A.3 : Compensation de mouvement par blocs superposés

Un pixel est alors reconstruit comme la somme pondérée de tous les pixels de l'image précédente et qui sont déplaçés à sa position (voir la figure A.3).

Puisque ce concept de blocs superposés réduit l'effet de bloc, l'erreur de reconstruction ne contient plus de hautes fréquences dans les transitions entre les blocs et peut être codée globalement. Ohta et Nogaki (1993) proposent de coder cette erreur avec un codage par sous-bandes.

Par ailleurs, plusieurs chercheurs proposent d'effectuer un post-traitement sur les images reçues au récepteur (I_r). Par exemple, Nakajima et al. (1994) modélisent ces images comme étant la somme des images originales I et d'un bruit blanc gaussien n. Ils utilisent le résultat de Lee (1980) prouvant que l'estimation $\hat{I}$ qui minimise

l'erreur quadratique moyenne peut être exprimée selon l'équation:

$$\hat{I} = E[I_r] + \frac{\sigma_{I_r}^2 - \sigma_n^2}{\sigma_{I_r}^2}(I_r - E[I_r]) \tag{A.23}$$

où $E[I_r]$ et $\sigma_{I_r}^2$ sont la moyenne et la variance de I_r estimés dans un voisinage de (x, y) et σ_n^2 est la variance du bruit dans ce voisinage estimée à partir des statistiques locales de l'image reçue. Les deux termes suivants sont retenus pour ce bruit:

- l'activité du bloc estimée par sa variance σ_b^2: l'erreur est en générale plus grande aux contours spatiaux:
- des statistiques locales autour du pixel: le bruit est plus perçu dans les régions d'intensité uniforme.

A.3 Compensation de mouvement par région

A.3.1 Compensation par régions polygonales

Dans ces techniques, plusieurs blocs sont réunis pour former des régions. La décision d'unir ou de diviser des régions se fait pour optimiser un critère. Les critères les plus souvent utilisés dans la littérature sont les suivants:

A.3.1.1 Ressemblance des vecteurs de déplacement

La décision d'attribuer un bloc à une région se fait en se basant sur la différence entre le vecteur de déplacement du bloc et la moyenne des vecteurs de la région.

Bartolini et al. (1994) proposent de faire la segmentation en utilisant un algorithme d'accroissement des régions séparément sur les vecteurs de vitesse et sur les erreurs de reconstruction. Inversement au signal d'erreur les vitesses sont codées sans perte. Une condition nécessaire pour attribuer deux blocs à la même région de vitesse est donc qu'ils aient le même déplacement. Pour chacune des deux segmentations (des vecteurs de vitesse et des erreurs), on transmet la moyenne[5] et la frontière de chacune des régions. Ces frontières sont compressées avec un codage par chaines. Pour diminuer la quantité d'information à transmettre, ils codent dans le domaine spatial les régions dont l'erreur de reconstruction par compensation de mouvement est grande (on ne transmet pas le vecteur de déplacement pour ces régions). La compression obtenue avec cet algorithme reste insuffisante pour des applications à bas débit. En effet, les informations de segmentation à transmettre sont très volumineuses.

Carpentieri et Storer (1992) et Carpentieri et Storer (1994) ont implanté une architecture parallèle composée d'un processeur par bloc. A l'émetteur comme au récepteur, les processeurs voisins s'associent ensemble si leurs déplacements calculés sur l'image au temps précédent se ressemblent. Une mise à jour de cette segmentation est effectuée en se basant sur le déplacement au temps courant et est transmise au

[5]Dans le cas des déplacements la moyenne correspond exactement au vecteur de tous les pixels de la région (codage sans perte)

récepteur sous forme de liste de subdivisions. En effet, si le déplacement de deux blocs appartenant à la même région est assez différent, on décide de subdiviser cette région et on rajoute cette information dans la liste de subdivisions.

A.3.1.2 Erreur de reconstruction (EQM)

On subdivise une région si l'erreur de reconstruction qui lui est associée est grande. Pour fusionner des régions, on se base sur le critère précédent (ressemblance des vecteurs de déplacement). Des arbres sont souvent utilisés pour représenter la segmentation obtenue. Un exemple d'un arbre binaire est donné à la figure A.4. On peut alors construire l'arbre avec un algorithme "grossier à fin et fin à grossier" (Leonardi et Chen 1994) ou en commençant tout simplement à un niveau intermédiaire et en n'itérant qu'une seule fois (Chan et al. 1990).

Une fois l'arbre construit, Leonardi et Chen (1994) codent les vecteurs de déplacement des feuilles de manière prédictive. La prédiction se fait spatialement (par rapport à un bloc voisin) ou temporellement (par rapport au même bloc une image plus tôt). On choisit, entre les deux, celle qui donne la plus petite erreur de prédiction. Un drapeau doit être transmis pour indiquer le mode de prédiction utilisé. Le gros problème avec leur approche provient du fait qu'ils calculent toujours le mouvement par rapport à la même image de référence. Ceci cause une accumulation des erreurs au cours du temps. Il faut donc transmettre une image de mise à jour de temps en temps.

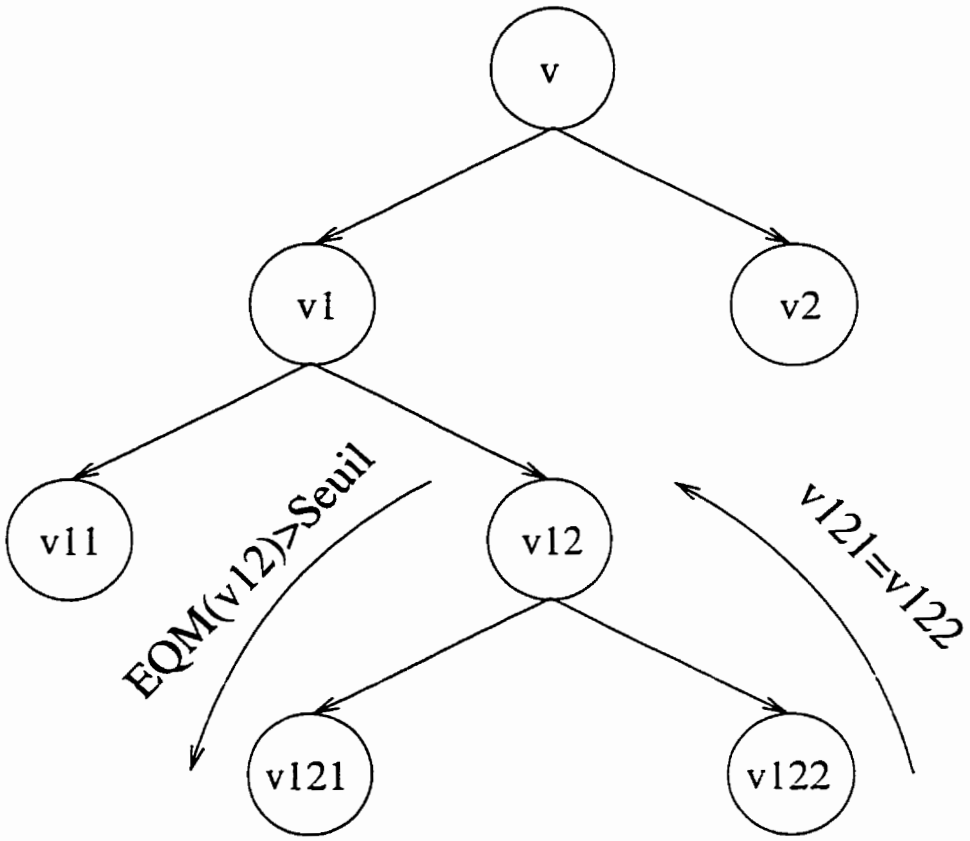


Figure A.4 : Exemple d'un arbre binaire utilisé pour représenter une segmentation

A.3.1.3 Rapport taux/distorsion

Les deux critères précédents effectuent la segmentation sans tenir compte de l'ajout d'information de segmentation à transmettre à chaque fois qu'on subdivise une région. Lee (1995) se base plutôt sur une fonction de coût qui tient compte de la qualité des images et de la compression obtenue. La fonction de coût à minimiser est la suivante:

$$CT = \sum_{b \in B} C(b) = \sum_{b \in B} D(b) + \lambda R(b) \tag{A.24}$$

où B est l'ensemble des blocs de taille variable, $D(b)$ est l'erreur de reconstruction associée au bloc b, $R(b)$ est le nombre de bits requis pour le codage de ce bloc et λ est un multiplicateur de Lagrange.

La décision de diviser un bloc en quatre ou de fusionner quatre blocs en un seul sera donc prise si elle apporte une diminution à la fonction de coût CT.

Ces approches sont des compromis sur les approches basées-bloc. La forme des régions est limitée à des unions de blocs et le mouvement de ces régions est représenté par un modèle translationnel (section A.1.6). Elles ne font en définitive qu'essayer de diminuer l'effet de bloc.

A.3.2 Compensation par régions à modèle de mouvement

A.3.2.1 Segmentation spatio-temporelle

Dufaux, Moscheni et Lippman (1995) proposent de raffiner la segmentation spa-

tiale obtenue. Ils fusionnent les régions adjacentes ayant des paramètres de mouvement proches et segmentent davantage les régions ayant une erreur de reconstruction grande.

D'autre part, Zhang, Bober et Kittler (1995) effectuent deux segmentations: spatiale et temporelle. La localisation de la segmentation temporelle sera alors améliorée en plaçant chacune de ses arêtes à la position de celle dans le domaine spatial qui lui est la plus proche.

D'autres chercheurs (Bonnaud et Labit 1994, Pardàs et al. 1994, Salembier, Gu, Pardàs et Kunt 1994, Salembier et al. 1995) optent plutôt pour une segmentation spatio-temporelle. Ils initialisent la segmentation dans le domaine spatial et l'adaptent à chaque itération en déplaçant les contours des régions en se basant sur les paramètres de mouvement déjà estimés. Dans le cas de Bonnaud et Labit (1994), les frontières des régions sont en plus mises à jour pour se rapprocher des zones de fort gradient des intensités lumineuses en se basant sur des transformations de type "snake" (Kass, Wikin et Terzopoulos 1988).

A.3.2.2 combinaison de la résolution du problème de la segmentation et de l'estimation du mouvement

Bouthemy et Francois (1993) optent pour minimiser la différence entre le mouvement estimé par le modèle ($\vec{\hat{v}}$) et le vrai flux optique ($\vec{v}$). Pour ne pas avoir à calculer directement le flux optique, ils minimisent l'énergie suivante obtenue en se basant sur

la contrainte du gradient (équation 1.2 qu'on écrit sous la forme $-\vec{\nabla}I(x).\vec{v} = \frac{dI}{dt}$):

$$U = \vec{\nabla}I(\vec{\hat{v}} - \vec{v}) = \vec{\nabla}I.\vec{\hat{v}} + \frac{dI}{dt} \tag{A.25}$$

De leur côté, Zheng et Blostein (1995) minimisent la longueur de description des informations à transmettre au récepteur[6] (MLD[7]). La fonctionnelle peut être décrite selon l'équation suivante:

$$\min_{\theta(n),N,b(n)} \sum_{n=1}^{N} [-\log_2 P_{\theta(n)}(e_{b(n)}) + LD_{\theta(n)} + LD_{b(n)}] \tag{A.26}$$

où N est le nombre d'objets, $e_{b(n)}$ est l'erreur de reconstruction de l'objet ayant $b(n)$ comme frontière. Cette erreur est modélisée selon une distribution gaussienne. Le premier terme de la sommation correspond à la contribution de l'erreur de reconstruction à l'entropie totale des informations à transmettre, le deuxième est l'entête nécessaire pour la transmission des paramètres du modèle de mouvement $\theta(n)$ et le troisième est l'entête nécessaire pour les frontières des objets représentées avec un codage sans perte par chaîne différentielle.

Ils ont utilisé une procédure d'accroissement des régions pour minimiser cette fonctionnelle. On fusionne alors deux régions quand cette union apporte une minimisation de la fonctionnelle. Il est à noter que la segmentation trouvée en effectuant cette minimisation n'est pas nécessairement cohérente avec la segmentation qu'aurait

[6] La segmentation et les paramètres décrivant le mouvement
[7] Minimum de la Longueur de Description

effectué le système visuel humain, par contre, elle est optimale pour le codage.

A.4 Compensation dense

A.4.1 Transmission de l'erreur de reconstruction

Les techniques récursives nous permettent d'estimer un champ dense de vecteurs au récepteur. Ce dernier utilise ce champ pour reconstruire les images avec une des méthodes d'interpolation présentées à la section 4.4. L'émetteur transmet donc uniquement l'erreur de reconstruction. Plusieurs chercheurs proposent de transmettre cette erreur uniquement pour les pixels en mouvement. Plusieurs algorithmes sont proposés pour la détection de ces pixels. Biemond, Looijenga, Boekee et Plompen (1987) proposent de transmettre l'erreur de reconstruction uniquement pour les pixels en mouvement avec la position de ces pixels dans l'image. La détection de ces pixels se fait en comparant la différence pixel par pixel des images successives avec un seuil. La vitesse des pixels statiques est alors remise à zero pour les adaptations futures. D'autre part, Walker et Rao (1984) transmettent tous les pixels avec un drapeau spécial pour indiquersi le pixel est à déplacement nul ou s'il faut estimer son déplacementà partir des déplacements de ses voisins à l'étape précédente.

Namazi, Penafiel et Fan (1994) et Matthews et Namazi (1995) proposent de détecter les pixels en mouvement par l'intermédiaire du test statistique suivant:

$$H_0 : \zeta(x,y) = I_{t+1}(x,y) - I_t(x,y) = 0 \quad (A.27)$$

$$H_1 : \zeta(x,y) = I_{t+1}(x,y) - I_t(x,y) = -\vec{\nabla} I_t(x,y).\vec{v}(x,y) \quad (A.28)$$

L'hypothèse H_1 signifie que le pixel est en mouvement, tandis que l'hypothèse H_0 correspond à un pixel statique.

Ils résolvent ce problème de classification en utilisant la méthode du rapport de vraisemblance:

$$\Lambda = \frac{P[\zeta(x)|H_1]}{P[\zeta(x)|H_0]} \begin{matrix} H_1 \\ > \\ < \\ H_0 \end{matrix} \eta \quad (A.29)$$

Pour trouver le seuil de décision η, ils posent les deux hypothèses suivantes:

$$f(\zeta(x,y)|H_0) = N(0,\sigma_\epsilon^2) \quad (A.30)$$

$$f(\zeta(x,y)|H_1) = N(-\vec{\nabla} I_t(x,y).\vec{v}(x,y),\sigma_\epsilon^2) \quad (A.31)$$

où $N(\bar{x},\sigma^2)$ est une distribution normale de moyenne $\bar{x}$ et de variance σ^2 et σ_ϵ^2 est la variance des erreurs dans l'acquisition des images.

Ils forment une image unidimensionnelle qui contient les pixels qui bougent. Les

deux composantes des vecteurs de mouvement de ces pixels sont codés séparément par transformation TCD. Pour exploiter les corrélations entre les vecteurs de mouvement des pixels adjacents, la difference entre les coefficients de transformation d'un pixel et d'une prédiction effectuée sur les coefficients des pixels adjacents est transmise. L'entropie de cette difference est plus petite que l'entropie des coefficients de transformation ce qui permet de la coder avec moins de bits.

Nous pouvons développer l'équation $I_{t+1}(x, y) = I_t(x - v_1, y - v_2)$ en série de Taylor autour de $(x - v_1^{\text{init}}, y - v_2^{\text{init}})$, pour trouver:

$$I_{t+1}(x, y) = I_t(x - v_1^{\text{init}}, y - v_2^{\text{init}}) + \vec{\nabla} I_t(x - v_1^{\text{init}}, y - v_2^{\text{init}})(\vec{v} - \vec{v}^{\text{init}}) - e(x, y, \vec{v}^{\text{init}}) \tag{A.32}$$

où $\vec{v}^{\text{init}} = (v_1^{\text{init}}, v_2^{\text{init}})$ est une estimation de $\vec{v} = (v_1, v_2)$ et $e(x, y, \vec{v}^{\text{init}})$ est l'erreur introduite par la linéarisation dans l'équation précédente.

L'erreur de reconstruction en se basant sur $\vec{v}^{\text{init}}$ est donc:

$$DTD(x, y, \vec{v}^{\text{init}}) = \vec{\nabla} I_t(x - v_1^{\text{init}}, y - v_2^{\text{init}})(\vec{v} - \vec{v}^{\text{init}}) - e(x, y, \vec{v}^{\text{init}}) \tag{A.33}$$

Banham, Brailean, Chan et Katsaggelos (1994) se basent sur cette équation et proposent d'utiliser le signal de DTD pour adapter l'estimation des vitesses au récepteur selon l'équation A.33. En effet, en négligeant les termes de second ordre dans le développement de Taylor $(e(x, y, \vec{v}^{\text{init}}))$, cette équation peut être écrite sous la

forme:

$$\vec{v} - \vec{v}^{\text{init}} = \frac{DTD(x, y, \vec{v}^{\text{init}})}{\vec{\nabla} I_t(x - \vec{v}_1^{\text{init}}, y - \vec{v}_2^{\text{init}})} \tag{A.34}$$

Ils proposent alors d'utiliser le DTD transmis au récepteur pour reconstruire les vitesses $\vec{v}$ à partir des vitesses estimées à l'itération précédente et de reconstruire les images en se basant sur ces vitesses. D'autre part, Huang et Zhuang (1995) proposent de se baser plutôt sur des vecteurs de déplacement transmis par bloc pour reconstruire le champ dense selon l'équation A.34.

Gísladóttir et Orchard (1994) proposent aussi de représenter le déplacement d'un pixel (x, y) comme étant la somme du vecteur de déplacement estimé pour son bloc par appariementde blocs (voir 1.3.1) $\vec{v}^{\text{init}}$ auquel on ajoute un vecteur de différence $\vec{v} - \vec{v}^{\text{init}}$. Ils proposent de décomposer ce dernier selon une pente et un module et de fixer une direction θ_b par bloc ce qui leur permet de trouver un module $d_\theta(x, y)$ par pixel correspondant à cette orientation en se basant sur l'équation A.34. Ces modules sont corrélées dans un même bloc et peuvent être codées efficacement par transformation TCD. Le choix de θ_b se fait pour minimiser la variance de $d_\theta(x, y)$. Malheureusement, le champ dense obtenu en se basant sur l'équation A.34 est très sensible au bruit à cause des deux raisons suivantes:

- le terme d'erreur $e(x, y, \vec{v}_b)$ n'est pas négligeable car $\vec{v}_b$ est en général loin du vrai mouvement (sauf au cas du mouvement translationnel fronto-parallèle):
- le calcul de $\vec{\nabla} I_t$ et de DTD est sensible au bruit.

A.4.2 Estimation et codage du mouvement en multirésolution

Moulin et Loui (1993) et Krishnamurthy et al. (1995) proposent une procédure "grossier à fin" pour estimer et coder le flux optique. Ce dernier est alors transformé à chacun des niveaux de la pyramide en utilisant les élements finis de Yserentant (1986) comme fonctions de base. A chaque niveau, les coefficients de transformation sont estimés pour minimiser la contrainte du gradient. Le mouvement estimé est, par la suite, projeté à un niveau plus fin et mis à jour pour minimiser la contrainte du gradient à ce niveau. Ils transmettent les coefficients de transformation du mouvement au niveau grossier et des vecteurs de mise à jour à chacun des autres niveaux.

Une autre méthode en multirésolution est proposée par Haddadi et Kuo (1994*a*) et Haddadi et Kuo (1994*b*). Ils divisent le flux optique, qu'ils supposent disponible, en multirésolution. Ils transmettent au récepteur le flux optique de la plus basse résolution et les mises à jour de ce flux à chacune des autres résolutions codées avec un codage sans perte. Une selection est effectuée sur les vecteurs de mise à jour à chaque résolution pour transmettre ceux qui apportent une amélioration significative à l'erreur de reconstruction. Le critère pour faire cette selection à un niveau l est la différence entre l'erreur de reconstruction à ce niveau et celle au niveau $l-1$. On transmet alors les vecteurs de déplacement uniquement aux endroits où cette différence est grande. Ils discutent de la possibilité d'utiliser le flux transmis à basse résolution pour reconstruire des images de basses résolutions au récepteur et de les

interpoler pour reconstruire les images originales ($\hat{I}_P$). Ils proposent de combiner l'image $\hat{I}_P$ avec l'image reconstruite par compensation du mouvement à partir du champ de mouvement dense et en utilisant l'interpolation bilinéaire expliquée à la section 4.4.1 ($\hat{I}_C$). La justification de ceci est que $\hat{I}_P$ est de bonne qualité là où le champ de luminance est doux (la variance des intensités σ_I^2 est petite). Par contre, $\hat{I}_C$ est de bonne qualité là où le champ de vitesses est doux ($\sigma_{\vec{v}}^2$ est petit). Ils se basent sur l'équation suivante pour réaliser cette combinaison:

$$\hat{I} = \frac{1-\sigma_{\vec{v}}^2}{2-(\sigma_{\vec{v}}^2+\sigma_I^2)}\,\hat{I}_C + \frac{1-\sigma_I^2}{2-(\sigma_{\vec{v}}^2+\sigma_I^2)}\,\hat{I}_P \qquad (A.35)$$

Cette combinaison donne plus d'importance à l'image de meilleure qualité entre $\hat{I}_P$ et $\hat{I}_C$ dans la reconstruction de l'image $\hat{I}$.

Annexe B

Guide d'utilisation du logiciel

Le logiciel CV-CMD a pour but de Coder des séquences Vidéo par Compression du Mouvement Dense approximé par le flux optique. Ce logiciel utilise le fichier *constantes.h* pour spécifier tous les paramètres de codage.

B.1 Installation du logiciel

La figure B.1 montre la structure du répertoire du logiciel CV-CMD. Tous les répertoires doivent et les sous-répertoire dans cette figures doivent être créer avant de compiler le logiciel pour assurer une bonne fonctionnalité de ce dernier. Les fichiers de code (`cv-cmd.c`, `weng.c`, `fdi.c` et `bsi.c`) et le fichier `makefile` doivent être présent dans la racine. Le répertoire `util` doit contenir tous les fichiers `*.h`. Avant d'utiliser le logiciel pour compresser une séquence ("rubik" par exemple), l'utilisateur doit faire les choses suivantes:

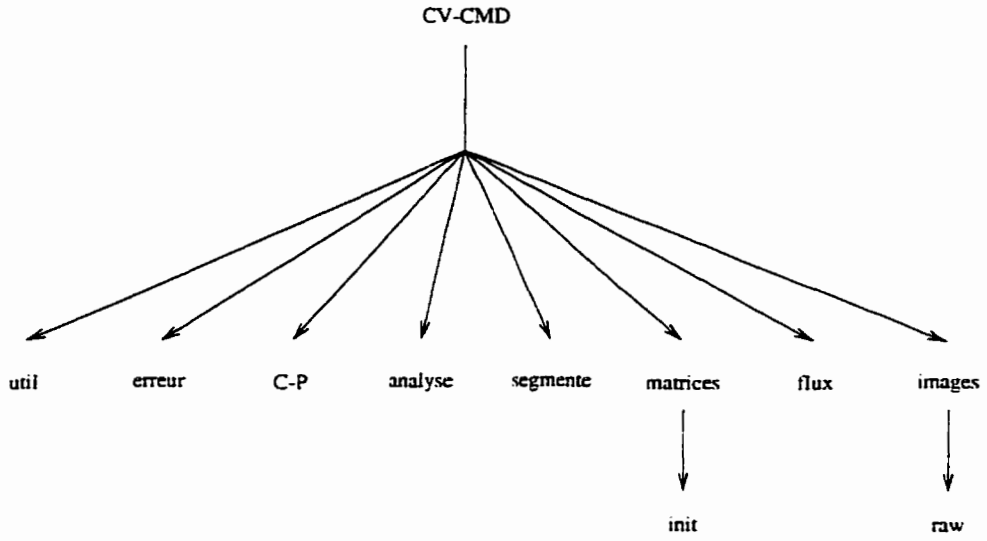


Figure B.1 : Structure générale du répertoire du logiciel CV-CMD

- Créer un répertoire qui s'appelle `rubik` dans chacun des sous-répertoires de CV-CMD (à part util et matrices/init). Un fichier de commande `creer-rep` est préparé pour cette fin. Il suffit de taper la commande

  ```
  $> creer-rep rubik
  ```

  et tous les répertoires nécessaires seront créés.

- Mettre les images de la séquence originale en format *sun raster* dans le répertoire images/rubik. Ces images doivent être appelées `rubik.1`, `rubik.2`, ...

- Ecrire dans le fichier `util/sequence.h` la commande `#include "rubik.h"`.

- Spécifier les paramètres de la compression dans le fichier `util/constantes.h`. Le contenu de ce fichier est expliqué à la section B.3. Il est à noter que lorsque vous exécuter une séquence pour la première fois, le paramètre `INIT_CALCULE` de ce fichier doit être mis à 0. Pour toutes les autres simulations exécutées sur

cette même séquence, ce paramètre peut être mis à 1.

- Compiler le logiciel en tapant tout simplement

  ```
  $> make
  ```

- Lancer le logiciel en tapant

  ```
  $> cv-cmd fichier_de_sortie
  ```

  Le fichier `fichier_de_sortie` contient tout l'affichage de sortie. Si l'usager écrit ecran pour ce fichier, l'affichage sera effectué à l'écran.

B.2 Fonctionnement du logiciel

Les figures B.2 à B.5 montrent le diagramme fonctionnel du logiciel.

B.3 Fichier *constantes.h*

Les constantes définies dans le fichier constantes.h sont les suivantes:

- `#include "dct.h"` Fichier qui contient toutes les constantes pour effectuer la TCD. Cette transformation est utilisée pour coder les images de mise à jour et l'erreur de reconstruction.

- `#include "sequence.h"` Fichier qui contient le nom et la taille de la séquence à compresser.

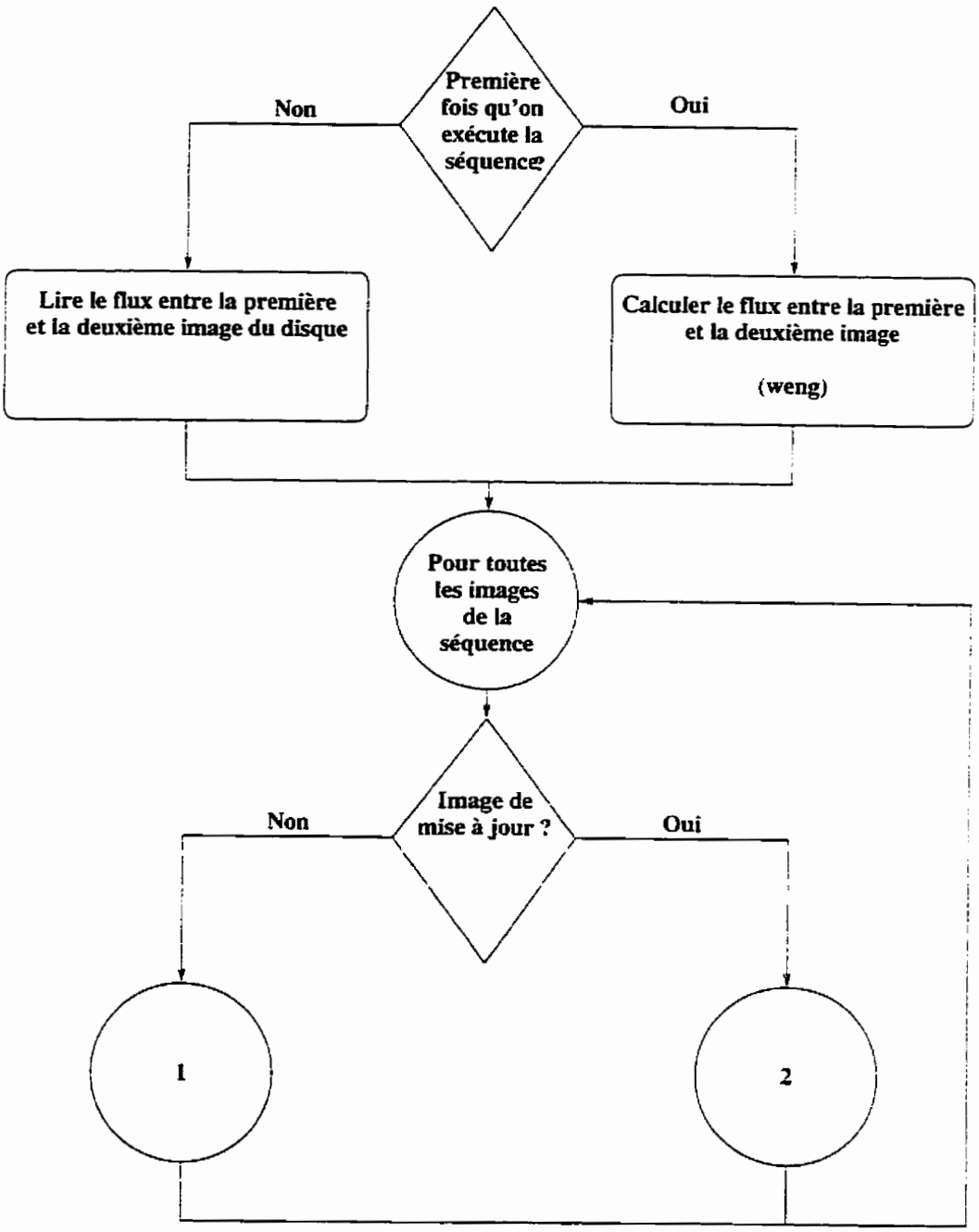


Figure B.2 : Diagramme fonctionnel du logiciel CV-CMD

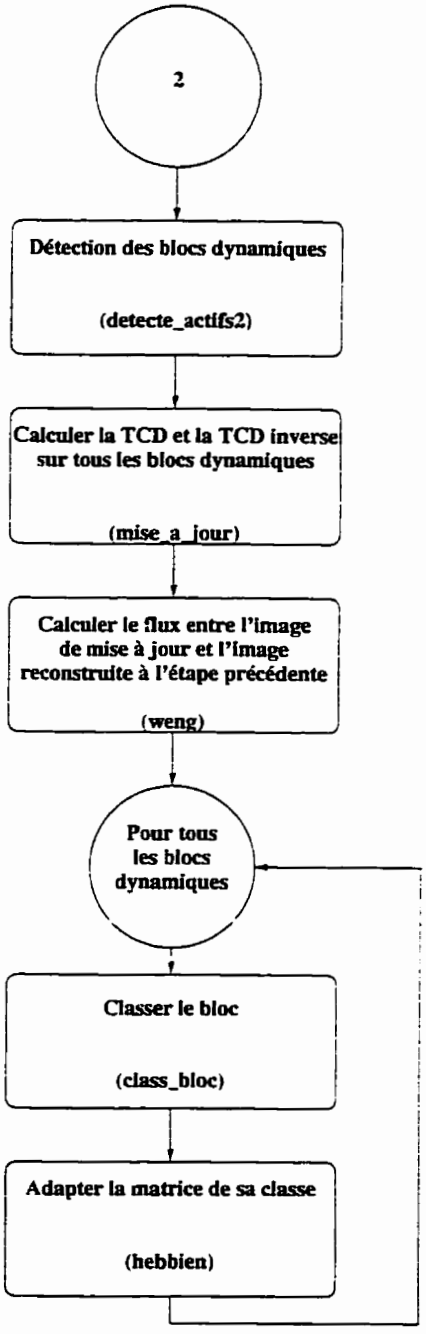


Figure B.3 : Diagramme fonctionnel du logiciel CV-CMD (suite)

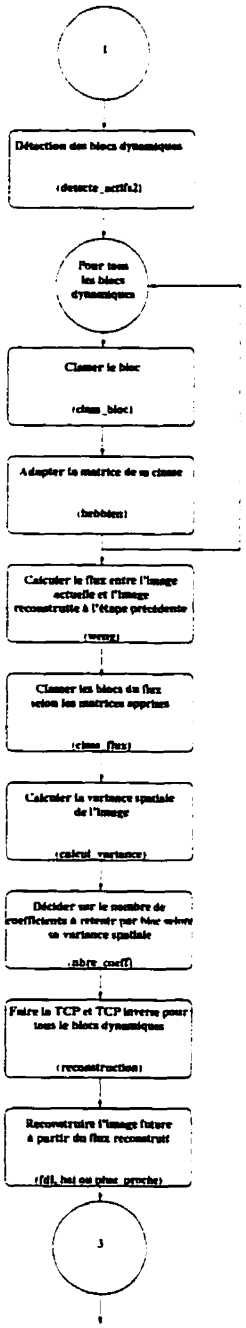


Figure B.4 : Diagramme fonctionnel du logiciel CV-CMD (suite)

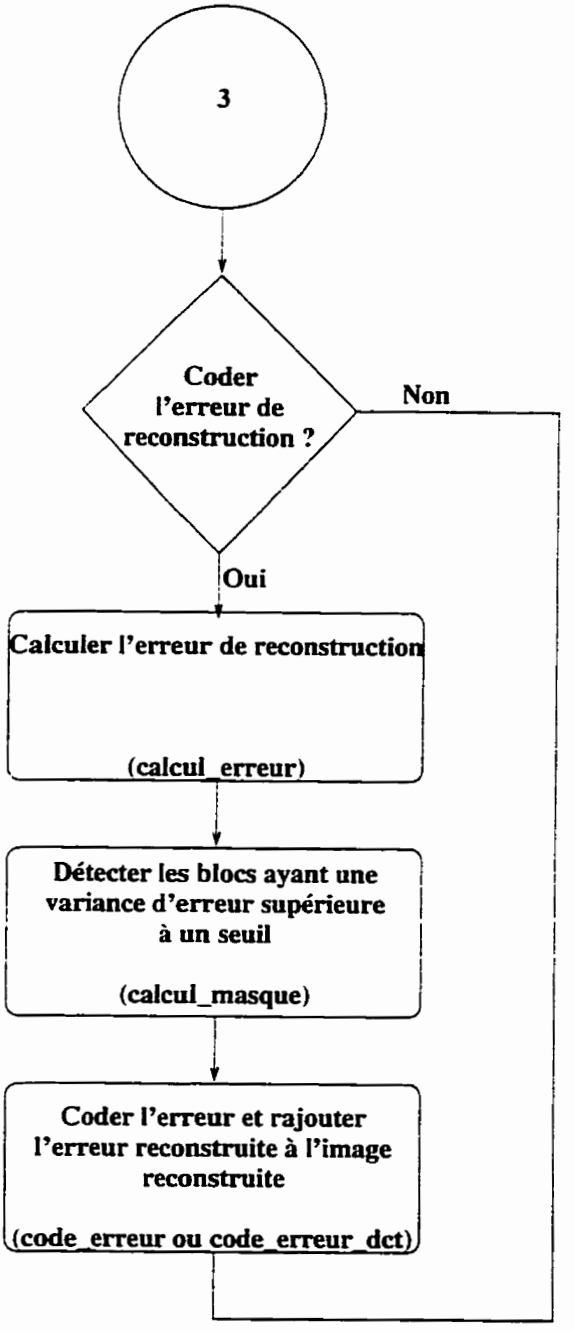


Figure B.5 : Diagramme fonctionnel du logiciel CV-CMD (suite)

- `#define BLOC_MAX 8` Taille des blocs à utiliser. Tous les tests ont été effectués sur des blocs de 8×8.

- `#define BLOC_CARRE (int)pow(BLOC_MAX,2.0)` Macro pour donner la taille des vecteurs à transformer décrivant l'amplitude et la phase. Cette taille est le carré de la taille des blocs.

- `#define NBRE_BLOCS_X (int)(MAX_X/BLOC_MAX)`

  et `#define NBRE_BLOCS_Y (int)(MAX_Y/BLOC_MAX)` Pour définir l'indice maximal des blocs en X et en Y respectivement.

- `#define NBRE_IMAGES 59` Nombre d'images dans la séquence.

- `#define NO_PREMIERE 1` L'indice de la première image de la séquence.

- `#define CONTINU 0` Si cette constante est à 1, nous continuons le codage à partir de l'image `NO_PREMIERE` sans réinitialiser le système. Cette option est pratique pour relancer une simulation quand le programme plante.

- `#define NBRE_CLASSES 7` Nombre de classes utilisées pour classer l'amplitude et la phase du mouvement.

- `#define NBRE_CLASSES_ERR 0` Nombre de classes utilisées pour classer le signal d'erreur de reconstruction. Dans cet exemple ce signal n'est pas transmis au récepteur.

- `#define CLASS_CP 0` pour décider si la classification est faite en se basant sur la composante principale (0) uniquement ou en faisant la projection sur toutes les composantes (1) (Sanger 1989).

- `#define CLASS_PREC 0` Faire la classification a partir du flux precedent (1) ou sur le flux actuel (0). Les expérimentations ont montrées que la classification doit être effectuée sur le flux actuel. Cette segmentation doit. par conséquent. être transmise au récepteur.

- `#define NBRE_COEFF_MIN 1` et `#define NBRE_COEFF_MAX 1` Spécifient la limite supérieure et inférieure du nombre de coefficients de transformation retenus pour l'amplitude et la phase. Le nombre de coefficients pour un bloc en particulier est choisi entre ces deux limites dépendemment de la variance spatiale de ce bloc. Dans cet exemple, les deux limites sont identiques et donc le même nombre de coefficients est retenus pour tous les blocs.

- `#define DELTA_MISE_JOUR 1000` Une image de mise à jour est transmise périodiquement à toutes les `DELTA_MISE_JOUR` images. Si cette période est supérieure au nombre totale d'images, ce critère ne forcera la transmission d'aucune image de mise à jour.

- `#define COUPURE_SCENE 20` A chaque fois que l'erreur quadratique moyenne sur les blocs dynamiques dépasse `COUPURE_SCENE`, une image de mise à jour est transmise.

- `#define NBRE_COEFF_MISE_JOUR 15` Nombre de coefficients à retenir spatialement de la TCD pour les images de mise à jour.

- `#define NO_INTERPOL 0`, `#define FDI 1` et `#define BSI 2` sont des méthodes pour faire la reconstruction des images à partir du mouvement. La première choisie tout simplement le pixel le plus proche, tandisque, les deux autre effectuent des interpolations pour reconstruire les intensité des pixels. `#define INTERPOL FDI` permet de choisir laquelle de ces méthodes nous voulons utiliser.

- `#define WENG 1`, `#define ANANDAN 2` et `#define BLOC 3` sont des méthodes pour calculer le flux optique. Les deux premières calculent le flux optique dense, tandisque, la troisième estime le mouvement par bloc. `#define METHODE_FLUX WENG` permet de choisir laquelle de ces méthodes nous voulons utiliser.

- `#define RECHERCHE 5` Définie l'espace de recherche pour l'estimation du mouvement par bloc.

- `#define DCT 1` et `#define KL 2` spécifie l'utilisation de la TCD où la TKL pour le codage de l'erreur de reconstruction tandisque, `#define NO_CODE 0` indique que cette erreur ne sera pas transmise. Ce choix est confirmé avec la constante `#define CODE_ERREUR NO_CODE`.

- `#define NBRE_COEFF_ERR 12` Si l'erreur de reconstruction est transmise, cette constante indique le nombre de coefficients à garder et `#define NBRE_BITS_ERR -1` indique le nombre de bits utilisés pour la quantification de ces coefficients. Le faite de mettre `NBRE_BITS_ERR` à -1 signifie que les coefficients ne sont pas quantifiés. Ceci est utilisé seulement pour des simulations.

- `#define CP_MAX_ERR 350` et `#define CP_MIN_ERR -150` indiquent la limite supérieure et inférieure du quantificateur des coefficients de l'erreur de reconstruction.

- `#define SEUIL_MASQUE_ERR 3.0` Si l'erreur de reconstruction est transmise, elle le sera uniquement pour les blocs ayant une variance d'erreur supérieure à `SEUIL_MASQUE_ERR`.

- `#define NBRE_BITS_AMP 5` et `#define NBRE_BITS_ANGLE 9` spécifient le nombre de bits utilisés pour la quantification des coefficients retenus de l'amplitude et de la phase respectivement. Si cette valeur est mise à -1, la quantification sera désactivée.

- `#define SEUIL_ACTIF 8` Seuil sur la RMS pour détecter les blocs dynamique. On choisit le plus petit entre ce seuil et la moyenne de la RMS à chaque itération comme expliqué à la section 4.3.1.

- `#define MAX_ECART 0` Si une classe n'est jamais utilisée au temps t. elle sera remplaçée, au temps $t+1$, par la classe la plus utilisée à laquelle on rajoute une valeur aléatoire. Cette constante indique la valeur maximale de cet aléatoire.

- `#define INIT_CALCULE 1` et `#define INIT_CALCULE_ERR 1` Indiquent si les matrices d'initialisation pour le mouvement et l'erreur de reconstruction respectivement après la première itération sont déjà calculés (1) ou s'il faut les calculer (0). Si ces matrices sont déjà calculé mais on veut les adapter encore plus, nous pouvons mettre la constante `APPREND_ENCORE` à 1.

- `#define ADAPT_MATRICES 1` Cette constante indique s'il faut adapter les matrices de transformation à chaque itération (1) ou garder ceux appris à la première itération (0).

- `#define SAUVEGARDE_MATRICES 0` Si cette constante est à 1. les matrices de transformation à chaque itération seront sauvegardées sur disque.

- `#define SEGMENTE_CALCULE 0` Pour des fins de simulation. la segmentation peut être faite manuellement. L'apprentissage des matrices de transformation sera, par conséquent, effectué à partir de cette segmentation.

- `#define NBRE_LOTS_INIT 50`, `#define NBRE_LOTS_FIN 2`, `#define NBRE_LOTS_ERR_INIT 20` et `NBRE_LOTS_ERR_FIN 10` Indiquent le nombre de lots d'apprentissage pour le mouvement et l'erreur de reconstruction respectivement pour la première itération d'initialisation et pour les autres.

D'autre part, `#define NBRE_LOTS_MJ 10` indique le nombre de lots pour les images de mise à jour. La constante `ECRETE_LOTS` nous permet de specifier à partir de quel lot le taux d'apprentissage ne sera plus diminué.

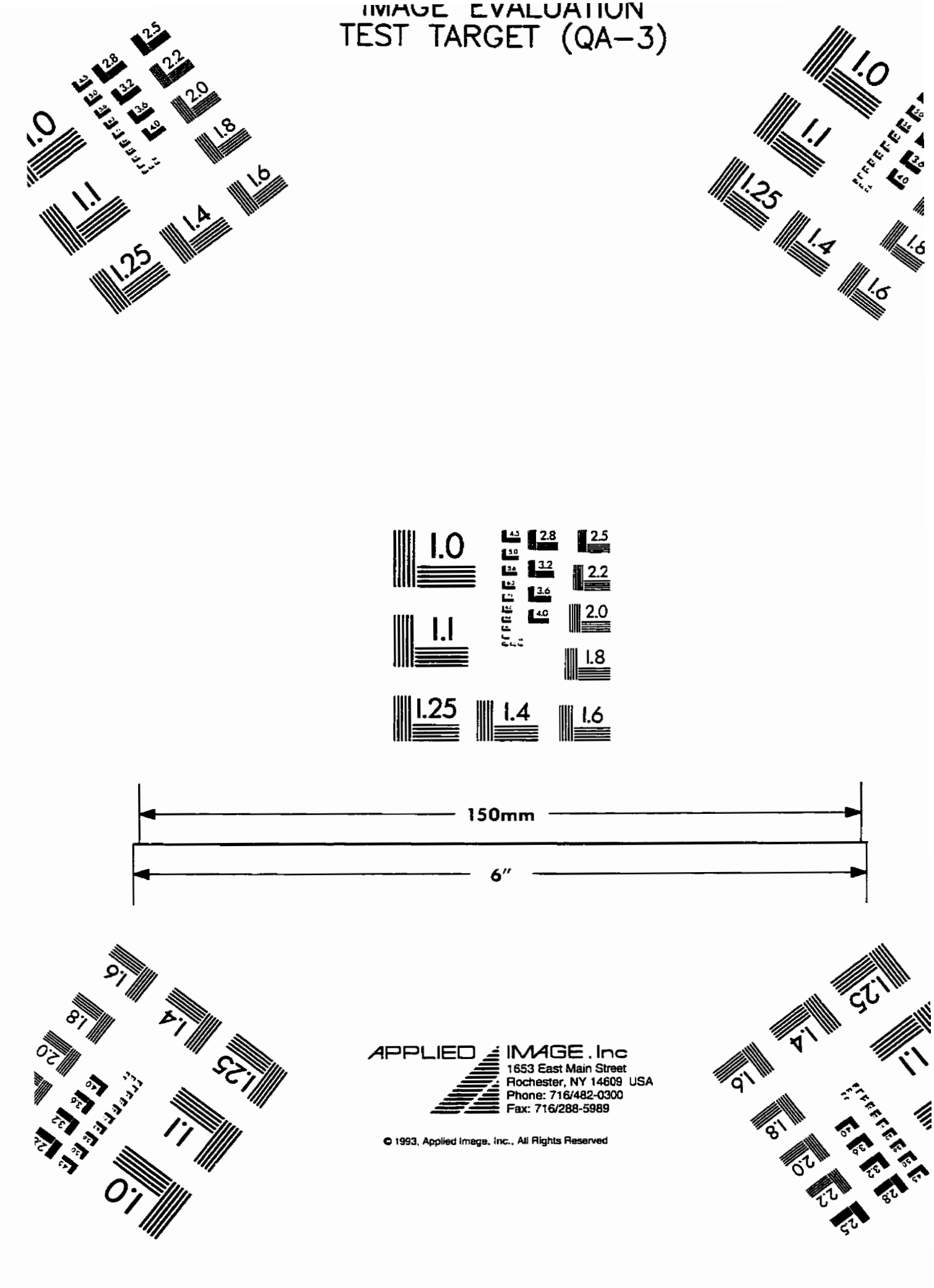


IMAGE EVALUATION
TEST TARGET (QA-3)
1.0
1.1
1.25
1.4
1.6
2.8
2.5
3.2
2.2
3.6
2.0
4.0
1.8
150mm
6"
APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989
© 1993, Applied Image, Inc., All Rights Reserved